3D Video Coding Based On Mixed Transform Techniques

A thesis submitted

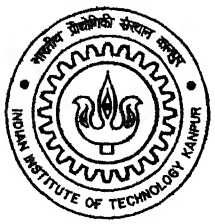
in partial fulfillment of the requirements

for the Degree of

Master of Technology

by

Ravi Kishore Paruchuru

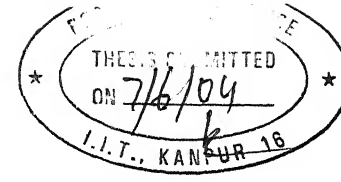


to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

June, 2004



CERTIFICATE

It is to certify that the work contained in the thesis entitled "**3D Video Coding Based On Mixed Transform Techniques**" by Ravi Kishore Paruchuru, has been carried out under my supervision and guidance. This work has not been submitted elsewhere for the award of any degree.

(Dr. Sumana Gupta)
Professor
Department of Electrical Engineering
Indian Institute of Technology
Kanpur, 208016

IIT Kanpur
June, 2004

Abstract

The thesis work is primarily concerned with the design of a "Complete" codec operating at low bit-rate (0.01-0.05 bpp). By "Complete" we mean it should possess advanced features of Scalability (spatial, SNR and Temporal) as well as Effective Packetization. The proposed codec uses both DCT and DWT in order to exploit the interframe and intraframe redundancies respectively. This approach has been adopted in view of the Mixed transforms performing better than single transform methods which are generally suitable for a particular subset of videos. A new Spatio-temporal tree structure has been devised for the unique frame structure that this proposed coder provides. Motion Compensation is not used in the coder. Comparisons of the proposed coder have been made with the existing codec that uses 3D-wavelet and 3D-SPIHT. Satisfactory results are obtained for low motion videos. To incorporate the features of Spatial and SNR scalability a layered bit-stream has been generated with multiresolution encoding. The packetisation of the video was done using Packetized zero tree wavelet coding algorithm. The packetized video from the proposed coder is tested for two types of errors : Random and Burst Packet errors. The results obtained indicate the robustness of the proposed coder against both Random and Burst packet losses as well as a graceful degradation of PSNR with increasing packet losses.

Acknowledgement

Foremost, I would like to express my sincere gratitude to my respected guide Dr. Sumana Gupta for her kind guidance that have helped this thesis work to assume a meaningful shape. I am grateful to her for providing me with references and her invaluable suggestions which helped me many a time when I was struck up with the problem in hand. A lot many thanks to her for giving me full freedom to think and approach in my own way and finally for carefully editing this thesis report.

I am deeply indebted to the faculty members who have taught me various signal processing & communication subjects, and clarified my doubts regarding the basics, and also helped in developing my view point.

I am thankful to Amir Said and William Pearlman for providing me with their coder which enabled me in comparing my work with that of theirs. Special thanks to Kausik Maiti for his constant suggestions regarding the finer details of the thesis.

I am also thankful to Dinesh, Prasad, Baju, Padlikar, Behera and Swagat for their help during my stay over here. Baju and Behera always helped me whenever my computer created problem.It will be very difficult to forget the happy, interesting momets with Bapi, Bhanu, krishna, Baju and Prabhakar who made my stay in IITK memorable. My heartfelt thanks goes to all of them.

Finally, I sincerely owe to my parents for their constant encouragement, support, blessing, guidance which have brought me to the portal of my present achievemet.

(*Ravi Kishore*)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Does Video Compression require any further research?

After the establishment of MPEG-2 in 1994, there were several voices stating that it was no more worth to put more effort into video coding. Because of the then achieved reduction to 1...2 % of the original data rate was considered as sufficient. Even at the initial stage of standardization of MPEG-4, more emphasis was put on incorporating additional functionalities rather than on increasing the compression ratio drew attention and finally with the definition of ACE(Adavanced Compression Efficiency) profile the coding efficiency could be increased by 30...50 % as compared to MPEG-2. Doubling the compression rate has also become the target of H.26L[1], the follow-up standard of H.263++ currently being developed within ITU-T SG16.

Today, when we already have huge amount of bandwidth available in our hand (as for example, today's glass fibers can carry terabits/sec, broadband Internet access via cable modems or xDSL is becoming widely available, today's DVDs and harddsics can store hours of video), when every ten years the cost of digital storage is getting

reduced by a factor of 100 or more (and this trend is expected to continue for coming next 20 years), a question naturally comes to our mind : why are we still continuing with R&D in video compression? The algorithms becoming more and more complex and the improvements are becoming more and more marginal. Then what is still missing today and what are the applications that demand for higher compression rates?

The answers to these questions can be debated among today's video coding research groups. However, we believe what is still missing is a universal full-featured video compression technique that will peform equally well for wide range of video coding applications.And the application field that still demands higher compression rates is mobile video services. Spectrum over the air is a scarce resource which cannot be enlarged. Even with the advent of UMTS/IMT 2000, the available data rate will be restricted to 144 kbits/s for mobile reception, higher rates up to 2 Mbits/s will only be available in microcells, i.e. in restricted areas like offices. It's a bare fact that 144 kbits/s is still a rather limited data rate for video transmission, if available coding schemes like MPEG-4 or H.263++ are used. On the other hand, mobile Internet traffic will grow at the same speed as Internet traffic in general (as the number of mobile terminals is exploding day by day). It is therefore foreseeable that the air bandwidth will remain a permanent bottleneck which justifies all efforts to squeeze video even futher.

1.2 Video coding: present scenario

Since the beginning of the 20th century, video coding has evolved from mostly an academic R&D field into a highly commercial business. The demand for image sequence coding has increased in leaps and bounds over last few decades mainly due to the factors like availability of personal work stations, Inernet multimedia and

also becuase of increased urge of information societies for better interaction among themselves. This in turn has motivated to build video coding standards. Standards have allowed easy interoperability across a wide range of equipments.

Since 1990, a series of video compression standards have been effected by two main standardization authorities : ISO/IEC and ITU-T. Each of these standards aim at different target applications. As for example, MPEG-1(standardized by ISO/IEC in the year 1991) is made for coding video at around 1.5 Mbps. Potential application area is video-CD. MPEG-2 (standardized by ISO/IEC in the year 1994) is optimized for HDTV. Target bit rate is 2-60 Mbps. Around 4 Mbps the quality of the image achieved is similar to PAL/NTSC/SECAM. MPEG-4(standardized by ISO/IEC in the year 1999) is basically for multimedia applications. It can code video at very low bit rate e.g. around 64 Kbps and finally H.26x, video compression standards by ITU-T are focussed to low bit rate video coding applications.

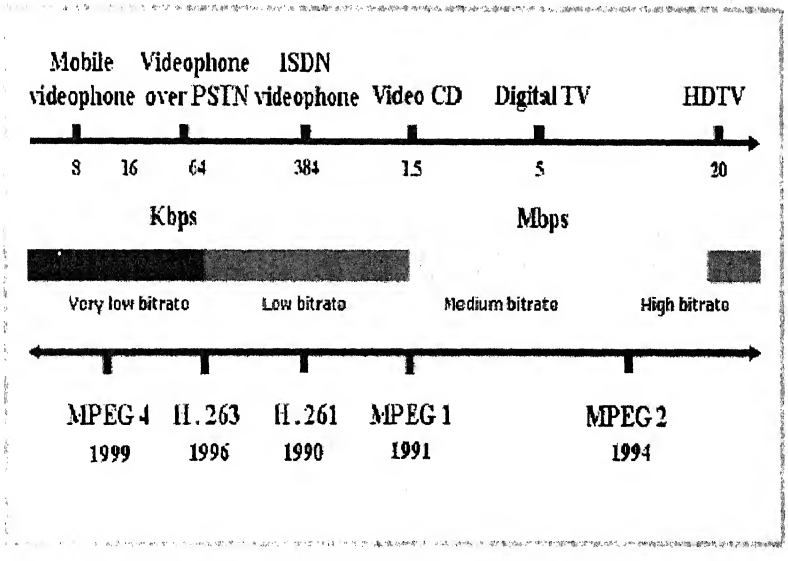


Figure 1.1: Potential Application Areas of Different Video Coding Standards

However different may these standards appear, all of them share a common coding strategy. They all use hybrid coding scheme (motion estimation and motion compensation followed by discrete cosine transform(DCT)). Now, as far as the coding

performance is concerned, this hybrid scheme seems quite satisfactory. Yet, it is not very perfect. It suffers from some serious drawbacks like high computational complexity, intense blocking artifacts and mosquito noise at low bit rates, possibility of infinite error-propagation. Last, but not the least, it provides inefficient support for scalable repressentation and coding, the most demanded feature of today's video coding.

Researchers have naturally been motivated to find out some alternative approaches. As a result, lots of non-standard techniques have been proposed in the last few years. Three dimensional (3D) transform coding is the most remarkable among them. (In this thesis, we shall confine ourselves mainly to this particular mode of video compression). As a part of "3D video coding" strategic movement, this 3D-DCT has been used to code group of consecutive video frames(GOF)[2]. This approach did not ultimately gain much appreciation mainly because of its heavy computational complexity. Also the blocking artifacts could not be eliminated due to independent block-by-block processing of video. However, 3D subband coding (SBC) using wavelets emerged as the most viable alternative in today's video coding standards.

At the early stage, 3D subband coding schemes have been designed and applied for mainly high or medium bit rate video coding. Karlsson and Vetterli[3] took the first step toward 3D-SBC using a simple 2-tap Haar filter for temporal filtering. Podilchuk, Jayant and Faravardin[4] used the same 3D subband framework without any motion compensation. It employed adaptive differential pulse code modulation(DPCM) and vector quantization to overcome the lack of motion compensation. Kronanader[5] first presented motion compensated temporal filtering within the 3D SBC framework. Ohm[6] and later Choi and Woods[7] refined the method and utilized different quantization techniques in application to subbands produced by perfect reconstruction filter banks.

In a parallel effort, in order to capture the multiresolutional nature of SBC schemes and to generate a scalable (both fidelity wise and resolution wise) bit stream, several scientists came out with their innovative ideas. Bove and Lippman[8] first proposed multiresolutional video coding with a 3D subband structure. Taubman and Zakhor[9] introduced a multi-rate video coding system using global motion compensation for camera panning, in which the video sequence was predistorted by translating consecutive frames before temporal filtering with 2-tap Haar filters.

However, it was only after 1993, When J.M. Shapiro[10] introduced Embedded zerotree wavelet(EZW) coder for still image coding, particularly when Pearlman and Said[11] launched their SPIHT (Set Partitioning In Hierarchical Trees) codec (one of the most successful improved EZW coders) in 1996, research on wavelet based visual information coding gained actual momentum. Lots of attempts have been made thereafter to extend the idea behind these two coders to exploit the temporal redundancy present in video signal. Chen and Pearlman[12] proposed 3D Improved embedded zero tree wavelet (IEZW) coder for video and showed promise of an effective and computationally simple video coding system without motion compensation. Another highly scalable embedded 3D SBC system[13] with tri-zero trees for low bit-rate environment was reported with coding results visually comparable, but numerically slightly inferior to H.263. Following the same line, Kim and Pearlman proposed 3D extension of SPIHT coder for low bit rate application[14]. They applied 3D dyadic wavelet to every GOF and coded the wavelet coefficients by 3D SPIHT. Even without any motion estimation and compensation this method performs measurably and visually better than MPEG-2. Finally in year 2002, a full featured, error resilient, scalable version of 3D SPIHT algorithm has been suggested by Sungdae Cho and WIllian A. Pearlman[16].

Today, the way research on 3D wavelet based video coding is progressing, it seems apparent that in near future, it will give birth to some new video coding standard

with 3D wavelet coding as its kernel.

1.3 3D video coding : pros and cons

3D Wavelet video coding was proposed with the notion to extend the wavelet decomposition so as to include the time domain as well. The reported advantages of 3D wavelet video coding include (1) symmetricity, (2) low computational complexity at both encoder and decoder end, (3) efficient scalability support, multiresolution transmission and display, (4) robustness of the generated bit-stream against transmission loss (due to absence of both spatial prediction as well as recursive loop in the codec architecture) and (5) absence of blocking artifacts.

Among the disadvantages, large storage space requirement at both encoder and decoder and longer processing delay are the main two points. Also, due to GOF-by-GOF processing, since the correlation between boundary frames cannot be taken into account, the PSNR drops by several dB at the GOF boundaries. This gives rise to temporal jerk while playing back the decoded video.

1.4 Motivation

Kim and Pearlman's 3D SPIHT coder[14] was able to draw appreciation because it achieved good objective as well as subjective quality (comparable with H.263) with minmal computational complexity. Noteworthy, no motion compensation was used there for coding low motion video. 3D dyadic wavelet was applied to decompose a GOF into several spatio-temporal subbands. 3D SPIHT was then used to quantize and code the wavelet coefficients. But, the point that is of our interest is , 3D SPIHT requires that the number of levels of decomposition along two spatial dimensions and the time dimension should be the same. Also, at the coarsest subband (refer

to Figure 1.2, there must be atleast 8 coefficients. This limits the flexibility of the codec. Suppose that the maximum allowable level of decomposition is two. However, this 2 levels of decomposition cannot fully exploit the inter- pixel redundancy present in any frame (let it have any standard size, be it QCIF or CIF). Thus, we feel that there is till scope for improvement as far as compression performance is concerned.

To cope up with the situation when there is considerable amount of motion either in the form of global camera motion or local object motion within a GOF, a motion compensation scheme was kept as an option. Now, needless to say, motion estimation itself is a comptatioanlly intensive task. Furthermore, the motion vectors (to be sent to the decoder end) simply increase the overhead information bits, which may count significant especially at very low bit rates.

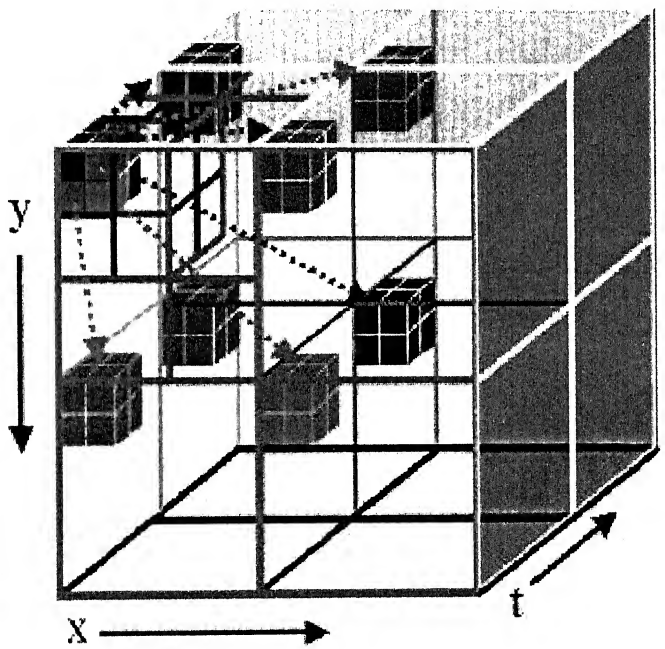


Figure 1.2: Demonstrating Parent Child Relationship used in 3D SPIHT Algorithm

Transform based image compression is a critical enabling technology at the heart of virtually all next generation computing and communication systems. In transform based compression schemes, the goal is to represent as much signal information with as few transform coefficients as possible. However, a particular transform is only

efficient for representing signals that are of the same class as the basis functions of the transform. Recently, Mixed transform coding techniques like[15, 17] have been described that overcome this limitation by building a signal representation from subsets of basis functions selected from multiple transform domains. Mixed transform coders have shown to consistently yield higher image quality than those based on a single transform for a fixed number of bits per pixel.

With these two points in mind, we have made an attempt to build a coding architecture that will be capable of performing satisfactorily at very low bit rates (also at higher bit rates) without using motion compensation at all. We maintain, it is the subjective quality and not the objective one that matters most. Because after all, the decoded video quality will be judged by some human being whose satisfaction depends solely on subjectve aspects of the video.

1.5 Organization of the thesis

The entire thesis is organized as follows. Chapter 2 describes the proposed video coding strategy and the key concepts involved in it. Chapters 3 a describe the Implementation details, Results and Discussions. Chapter 4 deals with the generation of a Layered bit-stream using Multiresolution encoding for providing Scalable bit-stream. Chapter 5 deals with packetizing the video using PZW coding and testing our video for two types of errors : Random and Burst Packet errors. Finally, we conclude and intimate the scope for future work in chapter 6.

Chapter 2

Very Low Bit Rate(VLBR) Video Coding

Very Low bit rate video coding is particularly important for video conferencing or video telephony type applications. In this chapter, we suggest a simple coding algorithm and show that it performs quite satisfactorily at very low bit rate (0.05 bpp) without requiring any motion compensation.

2.1 Some Observations

Before going into detail of our codec structure, let us view at different videos (that can be coded at very low bit rates) from slightly different angle. Figure 2.1 shows a collection of consecutive frames of a head and shoulder type video: "Miss America". This typical test video is widely used to check the performance of very low bit rate video coders. It possesses very good intra-frame as well as inter-frame inter-pixel correlation. To examine exactly how the temporal chnages occur, we have bisected the entire collection of frames along two planes, each being perpendicular to the video frames. Positions of these two planes have been adjusted so as to capture the

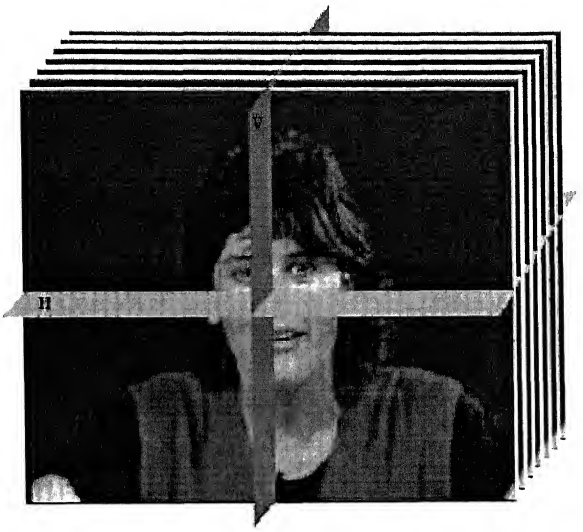


Figure 2.1: Miss America : A Typical Head And Shoulder Type Video

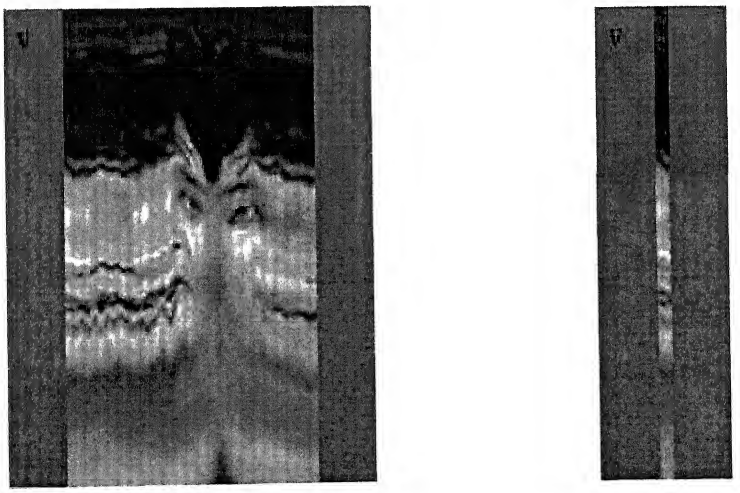


Figure 2.2: Side View : Temporal Changes Along Vertical Plane 'V' for 150 and 8 Frames

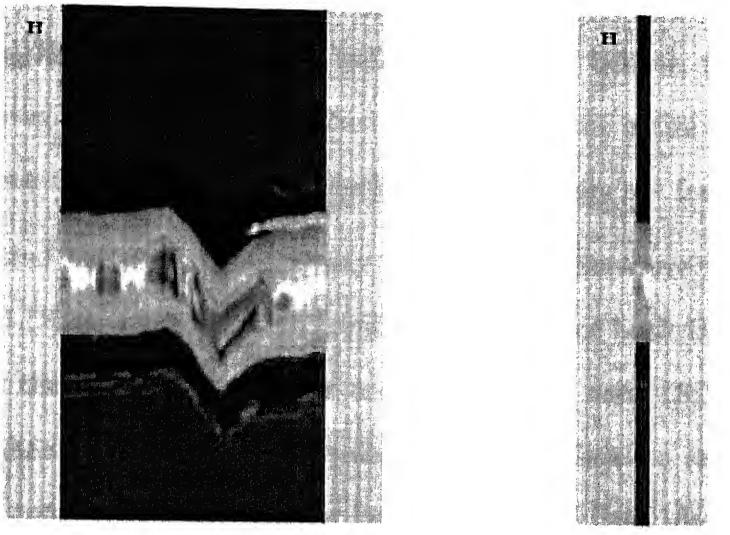


Figure 2.3: Vertical View : Temporal Changes Along Horizontal Plane 'H' for 150 and 8 frames

regions containing larger object movements. Figure 2.2 and Figure 2.3 display the side view and the vertical view respectively for collection of 150 and 8 frames (usual number of frames in GOF). Both of these figures clearly indicate the huge amount of temporal redundancy present among successive frames. Indeed, in most of the regions, except at the moving object-edges, there is no significant changes as such.

Same thing can be observed in case of any "head and shoulder" type video or any low motion video that is suitable for very low bit rate coding. There may be variety of textures, curves, edges etc. in the video frames, but , the change of those patterns with time is, in fact, significantly less.

2.2 Proposed Coding Strategy

In many of the recent papers on 3D subband video coding, subband filtering has been applied on the GOF both along the spatial dimensions and along the temporal dimension. The aim is to capture video information in different spatio-temporal subbands. Application of subband coding to individual video frames can be well

justified with our experience of wavelet based still image coding. However, use of the same along time axis can not be well justified. We believe, the advantage of temporal filtering can actually be realized if and only if the entire video is taken at a time as a consolidated information block, which is simply not feasible.

We have adopted the strategy that may seem to lie somewhere midway between 3D-SBC & 3D-DCT. The idea is to combine strengths of both DCT and SBC so as to improve the performance of our codec. We have used DCT to exploit the temporal redundancy present in the video signal. Needless to say, DCT has excellent energy compaction property for highly correlated data[18]. Also, DCT can play the role of motion compensation in more efficient way. In fact, it reduces the inter-frame redundancy among large number of consecutive frames[2]. (The actual number of frames depends on the number of frames in a GOF). To exploit the spatial redundancy, however, we have retained subband (wavelet) decomposition technique. The motivation has been obtained partly from the success story of wavelet for still image compression. Also, efficient high performance wavelet coders (e.g. SPIHT, SLCCA[19], EBCOT[20]), can be made use of. In particular, among many of the inherent attractive features of subband coding, we are basically interested in the following few.

- Since the subband(wavelet) filters operate on the entire frame, the blocking effect can be reduced to great extent.
- Subband coding provides us Scalability (both fidelity wise and resolution wise) as a free gift.
- Subband coding allows better bit rate control during encoding/decoding process.
- Subband coding is more robust under transmission or decoding errors. Because, errors on a particular subband may be masked by the information on

the other subbands.

2.3 Structure of the Proposed Codec

The block diagram of the encoder is shown is Figure 2.5. It includes four important operations. 1) Video Buffering, 2) Temporal Decorrelation, 3) Spatial Decorrelation 4) 3D-SPIHT for transmission.

Our coder accepts a pre-processed video. By pre-processing we mean noise reduction and/or temporal sub-sampling (sometimes, spatial sub-sampling, too) of a raw video data. This pre-processed video is buffered first. The length of the buffer is 2L, Where L is the number of frames in a GOF. The reason behind choosing a buffer of double length is to allow video buffering and video encoding to continue simultaneously and thereby to avoid unnecessary delay. DCT is then applied on the GOF along temporal dimension. This helps to decorrelate data along time. DCT, indeed compresses the energy toward the low frequency side and reduces the amplitudes of higher frequency components. Now, as there are L number of frames in a GOF, applying DCT along the time dimension will give rise to L number of different frequency planes. Let us call the first (lowest) frequency plane as DC plane and other frequecy planes as AC planes. The Figure 2.4 shows how DCT is applied and the resultant frame structure after DCT has been applied.

Each of the AC planes is then split into M (= 3N+1) number of subbands using N levels of dyadic wavelet decomposition (spatial decorrelation). However, in case of DC plane we adopt slightly different approach. We subtract the present DC plane from the previous DC plane and then apply wavelet decomposition on it too and then transmit the resultant frame structure using 3D-SPIHT with the help of a newly devised tree structure.The fine details of the tree structure will be explained in the next section. The advantage of 3D-SPIHT is that there is no need for any

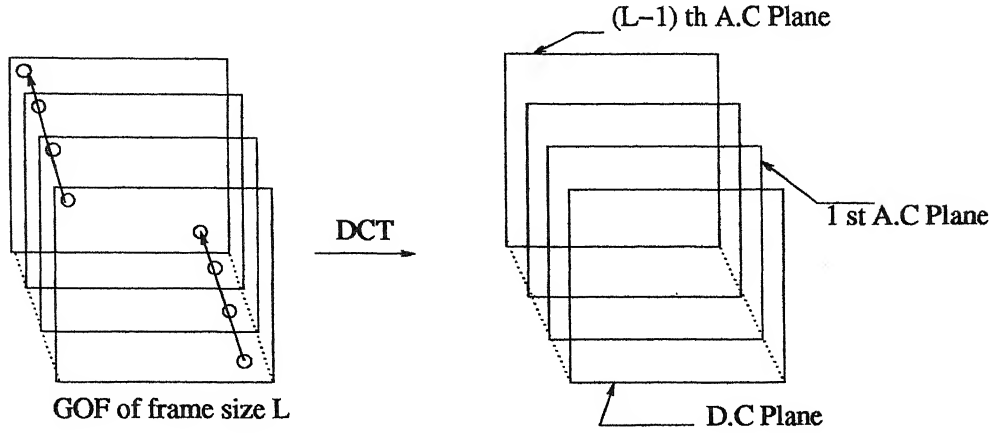


Figure 2.4: Frame Structure before and after application of DCT

seperate bit allocation. This sort of approach for the DC plane is carried out for all GOF's except the first one. For the first GOF the DC plane is also split along with other AC planes and the resultant frame structure is transmitted.

While, decoding we carry out the reverse procedure.

2.4 Key Operations and algorithms

In this section, we briefly give some preliminary ideas about the key operations and algorithms involved in the codec. We do not explain the details of DCT and Wavelet. This is available in at reference[21]. However, this section deals with motivation behind the SPIHT algorithm and the various factors that led to its development, and the mechanics of the new tree structure.

2.4.1 Motivation behind SPIHT

Wavelet transform tends to compact the energy of the input into a relatively small number of wavelet coefficients. For example, in naturally occurring images, much of the energy in the wavelet transform becomes concentrated into the LL_k subband. In addition, the energy in the high frequency bands (HL_i, LH_i, HH_i) is also con-

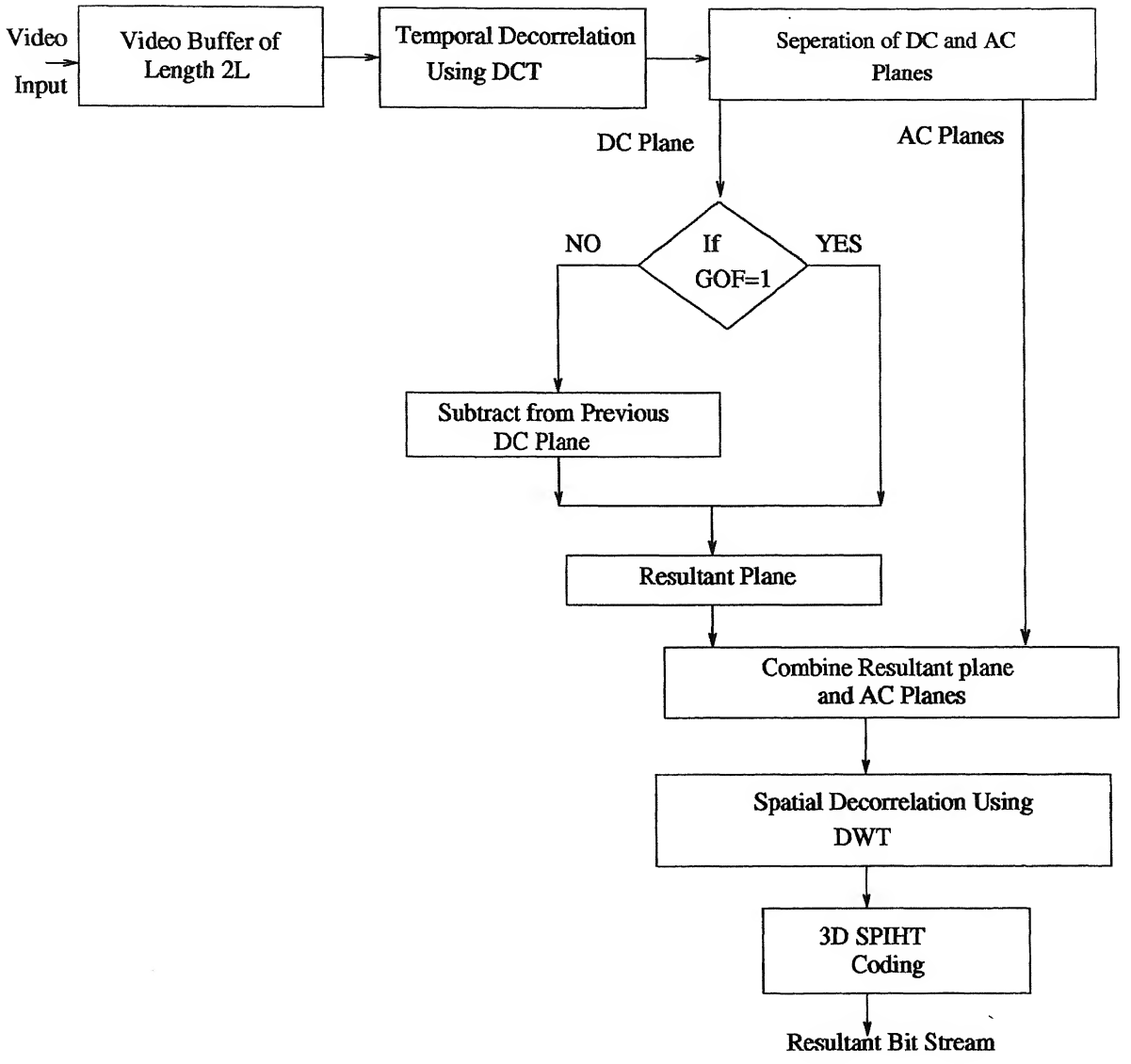


Figure 2.5: Block Diagram of Encoder

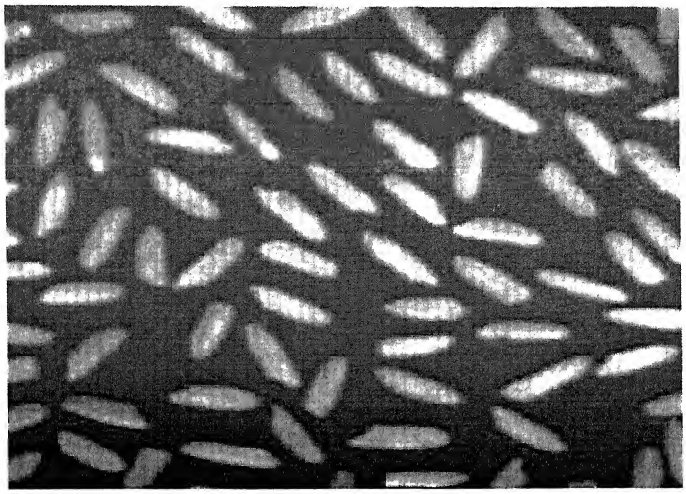


Figure 2.6: A Typical Test Image

centrated into a relatively small number of coefficients. Initially, designers of the wavelet coding systems recognised that low bit rate, low mean squared error (MSE) wavelet coders can be achieved by coding only few selected high energy coefficients. However, the problem was how to code both the position information as well as the magnitude information for each of these coefficients so that data can be recovered properly at the decoder end. In fact, depending on the method used earlier, the amount of resources required to code the position information sometimes turn out to be a significant fraction of the total, thereby negating much of the benefit of the energy compaction.

J. M. Shapiro[10] was the first person to show an efficient way to tackle this problem. In his pioneering work, he proposed a coder that simplifies the coding of the position informations of the significant wavelet coefficients using inter-band prediction (or prediction across frequency subbands) technique. The basic idea is to use the location of significant coefficients in one fequency band to predict the location and magnitude of significant coefficients in other fequency bands, thus reducing the cost of coding position information. Inter-band prediction technique that exploits the self similar, hierarchical nature of the wavelet transform, can easily be explained using

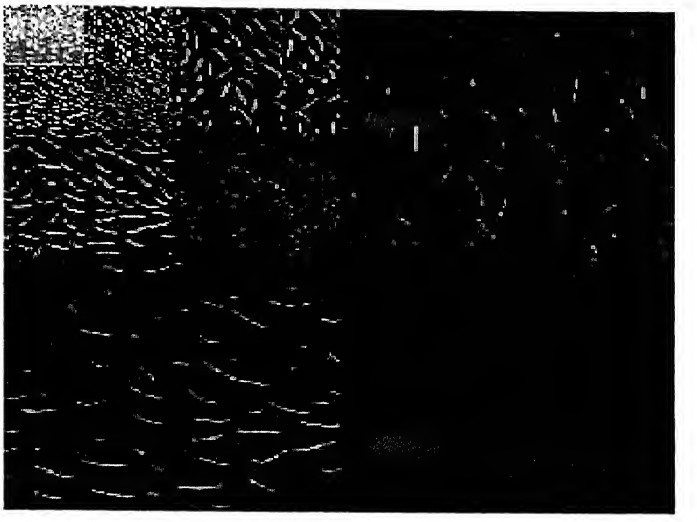


Figure 2.7: Wavelet decomposed image of test image shown in Figure2.6

Figure 2.7. Careful observation of this figure reveals that the significant coefficients in the high frequency subband do not occur at random locations, rather tend to cluster. Furthermore these clusters tend to occur at the same relative spatial locations in each of the high frequency bands which often correspond to discontinuities or edges that occur in the original image.This particular self similar relationship has been defined using "Spatial orientation tree" structure in[11].

2.4.2 Set Partitioning in hierarchical tree (SPIHT)

In this section, we briefly outline the concepts and operations involved in SPIHT coding. Interested reader is requested to refer to[11] for a more elaborate discussion.

As mentioned earlier that SPIHT coding utilizes the inter-band prediction technique that exploits the self-similar, hierarchical nature of the wavelet transform. Hence, at first, SPIHT algorithm defines a special tree structure called "Spatial orientation trees" (SOT) to perfectly represent the self similarity in case of image and "Spatio-Temporal orientation tree" in case of video using 3D wavelet using 3D-SPIHT for transmission (Figure 1.2) and Figure 2.8 shows how the "spatial orientation trees"

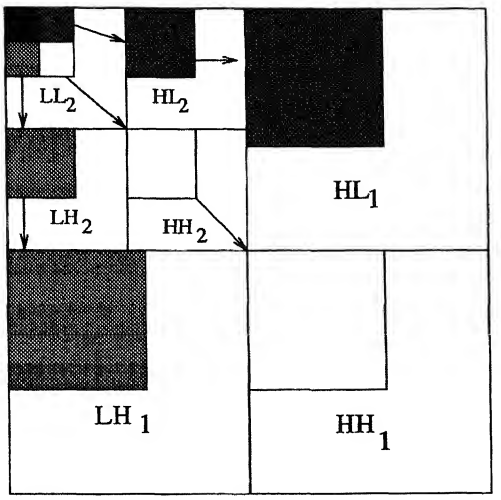


Figure 2.8: Parent-Off spring Dependencies in the Spatial Orientation Tree

are defined in case of a two-dimensional dyadic wavelet transform with two levels of decomposition. Each node of the tree corresponds to a wavelet coefficient and is identified by its coordinate. Its direct descendants (offspring) correspond to the coefficients of the same spatial orientation in the next finer level of the pyramid. The tree is defined in such a way that each node has either no offspring or four offsprings, which always form a group of 2X2 adjacent wavelet coefficients. However, their offspring branching rule is different, and in each group, one (e.g. one as indicated by a small black square in Figure 2.8) does not have any descendants. As the wavelet coefficients belonging to a SOT correspond to the same spatial location of original frame/image, it is very likely that if the magnitude of a coefficient in a certain node of a SOT does not exceed a given threshold value, then none of its descendents will exceed that threshold.

While encoding, SPIHT algorithm performs multiple scanning through the entire wavelet coefficient set. In each pass, it checks the magnitude of the coefficients against some threshold value. The initial threshold is computed as

$$T = 2^n where \quad n = \left\lfloor \log_2 \left(\max_{(i,j)} \{|c_{i,j}|\} \right) \right\rfloor \tag{2.1}$$

This threshold value is reduced to half everytime a pass is completed. The scanning order is always kept fixed. The members of the "list of insignificant pixels" (LIP) are scanned first. Then the members of the "list of insignificant sets of pixels" (LIS) and finally the elements of in the "list of significant pixles" (LSP) are scanned. Initially, LIP consists of the coordinates of all the wavelet coefficients in the coarsest subband, LIS consists of those of all SOT-roots and LSP is kept empty. While scanning the coefficients pertaining to LIP, if anyone is found significant (i.e with magnitude greater than or equal to the threshold value), a "1" bit and the associated sign bit are emitted. The corresponding coordinate is moved from LIP to LSP. Otherwise a "0" bit is sent. Next, while testing any set belonging to LIS, a "0" or a "1" bit is sent conveying the message whether all the elements in that tree are insignificant or not. If found not insignificant, the immediate four descendants of the current root of the tree are tested for significance. Any descendant found significant is moved to LSP with simultaneous emission of "1" bit and sign bit. And the significant descendant is placed in LIP and "0" bit is sent to the decoder. Once these four descendents are moved in this manner, the tree is treated as a "L"-type set. Previously, it is used to be treated as "D"- type set. The "L" type set is left inside for scanning second time in the same pass. It is to be scanned at the end. If however it turns out to be an empty set, it is completely eliminated from the scan-list. During scanning of the " L" set, as before, a "0" or "1" bit emitted from the encoder to inform the result of significance-test to the decoder. In case the set is found to contain some significant element, it is split into four "D"- type trees. The current root of this newly formed "D"- type set is one of the four immediate descendants of the previous root element. Scanning of LIP and LIS elements in this way is termed as "sorting".

In the "refinement" phase, the last phase of one pass, the n^{th} most significant bits of the coefficients that were entered into LSP in previous pass are sent as refinement bits. This way the coding process continues until the desired bit rate is reached.

In the decoder side, the same lists, LIP, LIS and LSP are prepared and updated in each pass. Also the same scanning pattern and decision/branching rule is used. As the input bits are read from the codestream, the decoder reconstructs the magnitude and the sign bits of the LSP members seen by the encoder. The coefficients of the final LIP and LIS sets are set to zero. Especially, in the wavelet transform of an image or a frequency plane, large sets of zero values exist which are identified efficiently by SPIHT with a single bit. Moreover, the significant coefficients are never represented by more bits than actually needed in their natural binary representation, since the highest "1" bit can always be known apriori in this process.

2.4.3 New Tree Structure

2.4.3.1 Why new tree structure?

There are a number of existing tree structures in literature like the tree used in 2D-SPIHT and the tree used in 3D-SPIHT. The tree used in 3D-SPIHT shown in Figure 1.2 cannot be used here because of the contrasting frame structure that this codec provides. However, tree used in 2D-SPIHT shown in Figure 2.8 can be used by transmitting the wavelet decomposed frequecy planes one at a time. But, using this tree structure would need a separate bit-allocation algorithm to distribute the bit-budget among different frequency planes. These bit-allocation methods are generally statistical in nature. Hence, they can fail sometimes or lead to non-optimum allocation of bits for each frequency plane many times. So, if we can device a new spatio-temporal tree for the unique frame structure that our coder provides we can use 3D-SPIHT algorithm, thereby avoiding the disadvantages

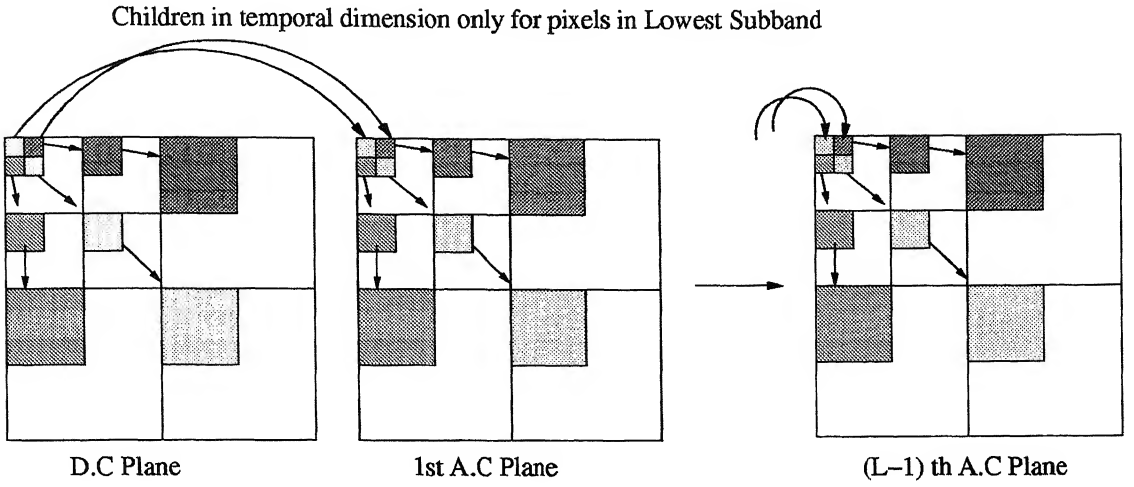


Figure 2.9: Tree Structure

involved in bit-allocation methods.

2.4.3.2 Mechanics of the tree structure used

The motivation behind this tree structure was the tree structure used in 2D-SPIHT. This tree structure is used for coding images and since we are working with image sequences (video) some changes would be necessary. It differs from the tree structure there in such a way that each of the pixels in the lowest sub-band in both the DC and the AC planes (except for the last AC plane) will have a child in the temporal dimension at the same spatial location in addition to the existing children of a normal 2D-SPIHT algorithm. Noteworthy, there are no children for the pixels outside the lowest sub-band in the temporal dimension. This new tree structure is explained in the Figure 2.9.

Chapter 3

Implementation, Results and Discussions

3.1 Important Implementation Details

The performance of the video coding systems with the same basic algorithm can be quiet different according to the actual implementation. Thus, it is necessary to specify the main features of implementation in detail. This section exactly deals with these things. In the following, we try to highlight on some implementation issues that we feel, greatly influence the performance of our coder.

- We have implemented the proposed codec in Matlab (Version 6.5.0.180913a Release 13). The videos used have a CIF resolution (352X288) at a frame rate of 30fps. However, we have restricted ourselves to gray video sequences.
- The first issue is the choice of the wavelet filter. We have tried with different orthogonal, compactly supported wavelet filters as well as bi-orthogonal, compactly supported wavelet filter pairs available in Matlab. Finally, we have decided to use Daubechies 9/7 bi-orthogonal wavelet filter pairs (both for

analysis and synthesis purpose) for fast motion videos because of its superior performance. By the term "superior" we mean that with the same wavelet quantizer, coder and decoder, among all other wavelet filters, Daubechies 9/7 bi-orthogonal wavelet filter pairs yield better quality (higher PSNR) of the decoded image. However, we have used "db2" filters for low motion videos since they themselves give better performance when compared to the existing methods. If we use Daubechies 9/7 filters in this case too, we can achieve even better performance, but at the cost of computational complexity.

- The second issue is the number of frames in each GOF. If we use very large number for frames in each GOF we can achieve better compression performance but at the cost of the extra delay and the buffer space needed for it. If we use very less number of frames in each GOF although we can save the buffer space it comes at the huge cost of loss in compression. So, there should be a trade-off and we found out this trade-off to be 8 after some experimentation.

- The peak signal to noise ratio (PSNR) calculated as

$$PSNR_{dB} = 10\log_{10}\left(\frac{255^2}{MSE}\right) \tag{3.1}$$

Where

$$MSE = \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(Frame_o[n,m] - Frame_r[n,m]\right)^2 \tag{3.2}$$

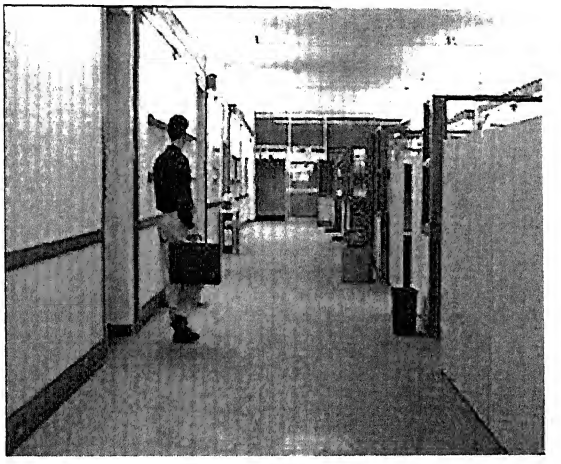
has been used as an objective measure of frame quality. In the Equation 3.2 Frame_O stands for an original frame and Frame_r denotes corresponding reconstructed frame. However, we emphasize that the true subjective quality of a reconstructed video-frame does not have any rigid connection with the frame PSNR. Also instead of inspecting the quality frame by frame, we should judge the video-quality as a whole, treating video as a three dimensional entity. Over and above, the actual evaluation of the reconstructed video quality ought

to be made by playing back the video with associated sound!

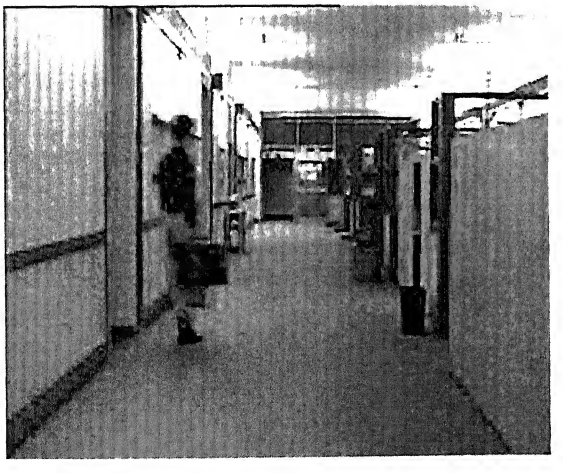
3.2 Results

We have tested different gray video sequences "Hall Monitor", "Akiyo", "Salesman" and "Carman". These video sequences represent a variety of object and camera motion. "Hall Monitor" is suitable for monitoring application. The background is always fixed and some objects(persons) appear and then disappear. "Akiyo" and "Salesman" are typical head and shoulder video sequences with relatively small object motion and stationary camera. These video sequences are suitable for video conferencing. "Carman" is a representative sequence for video-telephony application. This has some what more complicated object motion and a stationary camera. All the tests were performed at a frame rate of 30fps and CIF resolution.

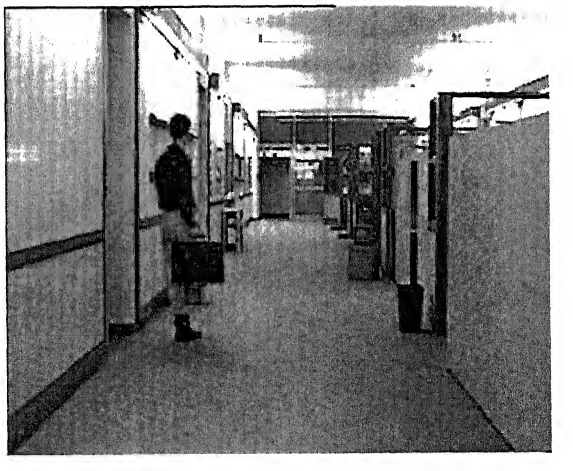
To see the visual performance of this coder at various bits per pixel we have shown few frames of various video sequences like "Hall Monitor", "Akiyo", and " Football". Figure 3.1 shows the reconstructed Hall Monitor 35^{th} frame at 0.02 bpp (60.825 Kbps) and 0.05 bpp (152 Kbps) and the Figure 3.2 also shows the reconstructed 100^{th} frame of Hall Monitor sequence at the corresponding bpp as for the earlier frame. Figure 3.3 shows the reconstructed 26^{th} frame of Akiyo video sequence at 0.01 bpp (30.41 Kbps) and 0.05 bpp and the Figure 3.4 shows the 39^{th} frame of the same Akiyo video sequence at the corresponding bpp as for the earlier frame. Figure 3.5 shows the reconstructed 3^{rd} frame of Carman video sequence at 0.05 bpp and 0.07 bpp (210 Kbps) and Figure 3.6 shows the reconstructed 36^{th} frame of the same Carman sequence at the same bpp as earlier. Our idea in showing the frames at various bits per pixel is to show how the visual quality of the reconstructed frame varies as the bit budget increases. Corresponding original frames are also shown.



(a) Original

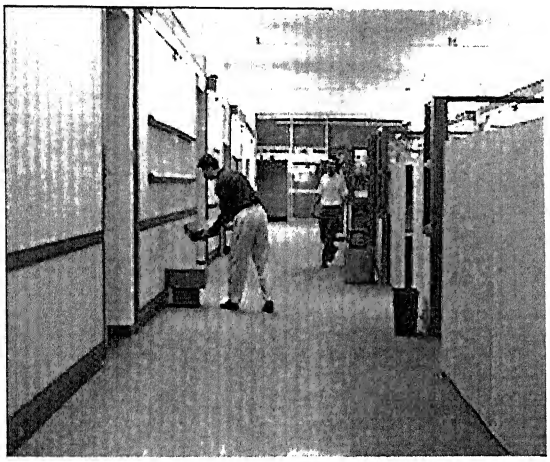


(b) Reconstructed at 0.02 bpp

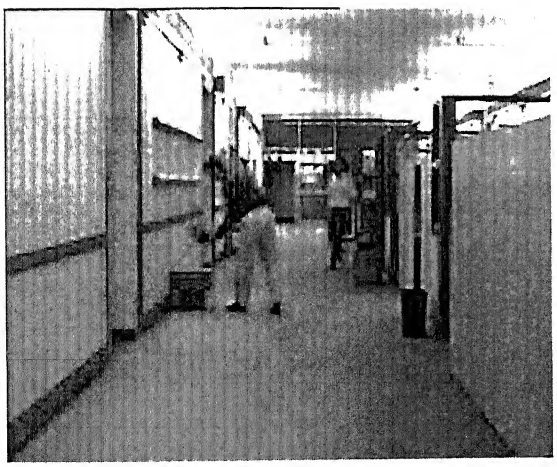


(c) Reconstructed at 0.05 bpp

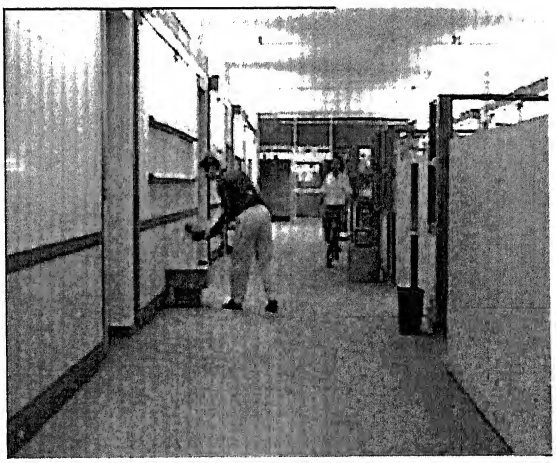
Figure 3.1: Hall Monitor : 35^{th} Frame



(a) Orginal



(b) Reconstructed at 0.02 bpp



(c) Reconstructed at 0.05 bpp

Figure 3.2: Hall Monitor : 100^{th} Frame

(a) Original

(b) Reconstructed at 0.01 bpp

(c) Reconstructed at 0.05 bpp

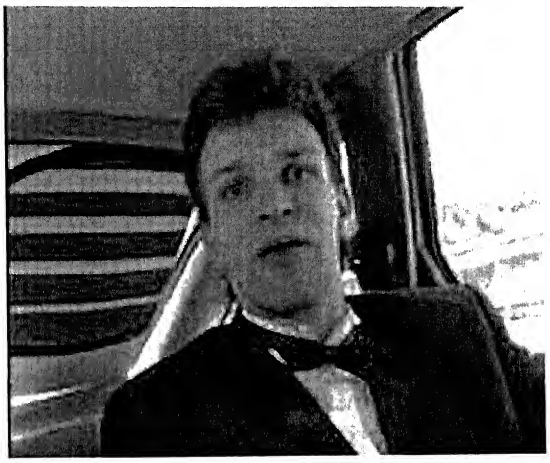
Figure 3.3: Akiyo Sequence : 26^{th} Frame

(a) Original

(b) Reconstructed at 0.01 bpp

(c) Reconstructed at 0.05 bpp

Figure 3.4: Akiyo Sequence : 39^{th} Frame



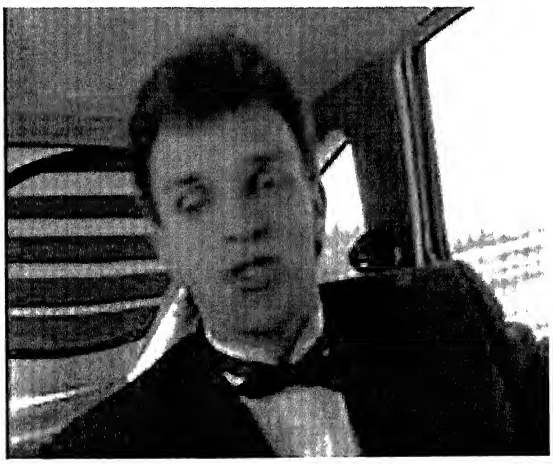
(a) Original

(b) Reconstructed at 0.05 bpp

(c) Reconstructed at 0.07 bpp

Figure 3.5: Carman Sequence : 3^{rd} Frame

(a) Original



(b) Reconstructed at 0.05 bpp

(c) Reconstructed at 007 bpp

Figure 3.6: Carman Sequence : 36^{th} Frame

3.3 Validation of the proposed coder

In this section we compare the performance of the proposed codec with the existing codec that uses 3D-wavelet for compression and 3D-SPIHT for transmission. This was downloaded from *http://www.cipr.rpi.edu/research/SPIHT/spiht3.html.* The paper[14] which describes this coder concludes that it performs better than H.263++ with all the advanced options in place when both PSNR and computational complexity are taken into consideration. So, if we can design a better coder than the 3D-wavelet coder in terms of both PSNR and computational complexity it automatically performs well in comparison with H.263++ which is considered good for video-conferencing and video monitoring applications. One difference with this coder is that the proposed coder uses 8 frames in each GOF and the 3D-wavelet coder that we have downloaded uses 16 frames in each GOF. If the number of frames in a GOF increase, the compression increases affecting adversely the delay and the buffer space. As the number of frames in a GOF decrease, the compression achieved also decreases, so also the delay factor and the buffer space required. Hence, there should be a trade-off between compression achieved and the delay factor, buffer space required. We have achieved this trade-off by taking 8 frames in each GOF. But, since we are comparing the proposed coder with the coder using 16 frames in GOF even if there are small losses in PSNR in comparison to the existing coder it can be taken as a better performance because of the above mentioned reasons.

Figures 3.7 to 3.12 show the frame by frame PSNR variation of the sequences Hall-monitor at 0.02 bpp and 0.05 bpp, Salesman at 0.01 bpp and 0.05 bpp and carman at 0.05 bpp and 0.07 bpp respectively. Table 3.1 summarises the frame by frame PSNR variations shown in the above figures. The table shows the mean PSNR values of the video sequences decoded at the corresponding bits per pixel for the number of frames shown in brackets in the video sequence column.

Video Sequence	BPP	1d DCT + 2d Wavelet	3d Wavelet	Gain
Carman (1-96)	0.05	28.02	28.33	−0.31
Carman (1-48)	0.07	30.26	30.50	−0.24
Hall Monitor (1-128)	0.02	30.09	27.07	+3.02
Hall Monitor (1-128)	0.05	35.72	32.33	+3.39
Salesman (1-128)	0.01	26.34	24.95	+1.39
Salesman (1-128)	0.05	32.76	30.94	+1.82

Table 3.1: Comparison Between proposed coder and 3D wavelet coder for various video sequences at different bpp

Now that the objective comparison has been observed (comparison of PSNR's) for various video sequences, we turn to the subjective comparison of the frames using the proposed coder and 3D wavelet coder. Figures 3.13, 3.14, 3.15, 3.16, 3.17 and 3.18 show some of the frames of the video sequences Hall monitor, Salesman and Carman. From the above mentioned figures of differnet video sequences, we can conclude that the motion blur in the proposed coder is comparatilvely lesser than the motion blur in 3D wavelet coder. We can observe this by seeing the moving parts of the video like man in the case of Hall Monitor sequence, hand in the case of Salesman sequence and face in the case of Carman sequence.

The disadvantage of this coder is that it does not work well for video sequences in which backaground changes rapidly. One example of such sequence is Flower garden. The reason for this is explained in order. The compression performance of DCT is at its best when the data being compressed has good correlation. Infact, it achieves same compression as KL transform (considered best transform for compression but very complex in comparison to DCT) when the correlation coefficient of the data being compressed is around 0.95. But, when the background changes rapidly this correlation falls down considerably leading to bad compression and thus bad performance of the proposed coder. Figure 3.19 shows the frame by frame PSNR variation of the above said video sequence and Figure 3.20 shows the PSNR gain/loss when compared to 3D wavelet coder.

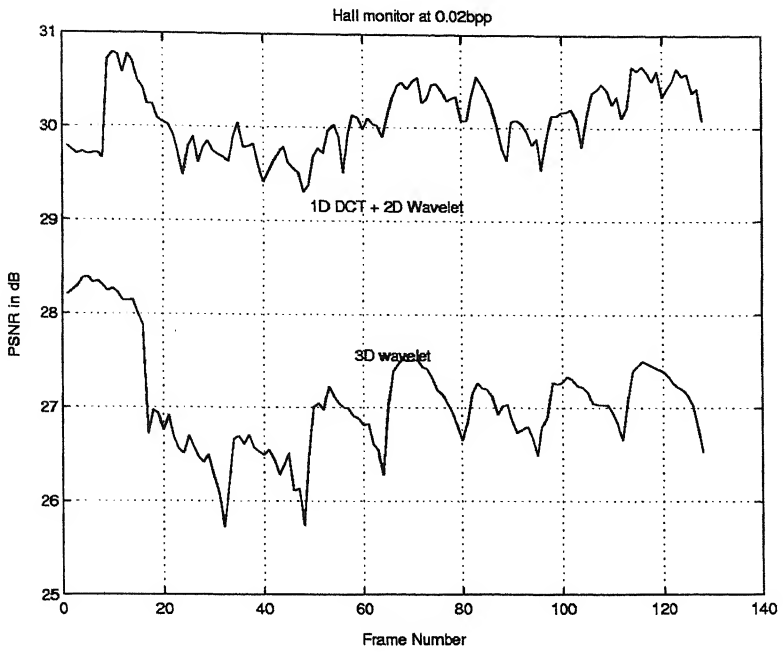


Figure 3.7: Comparison between the Proposed coder and 3D wavelet coder for Hallmonitor sequence at 0.02 bpp

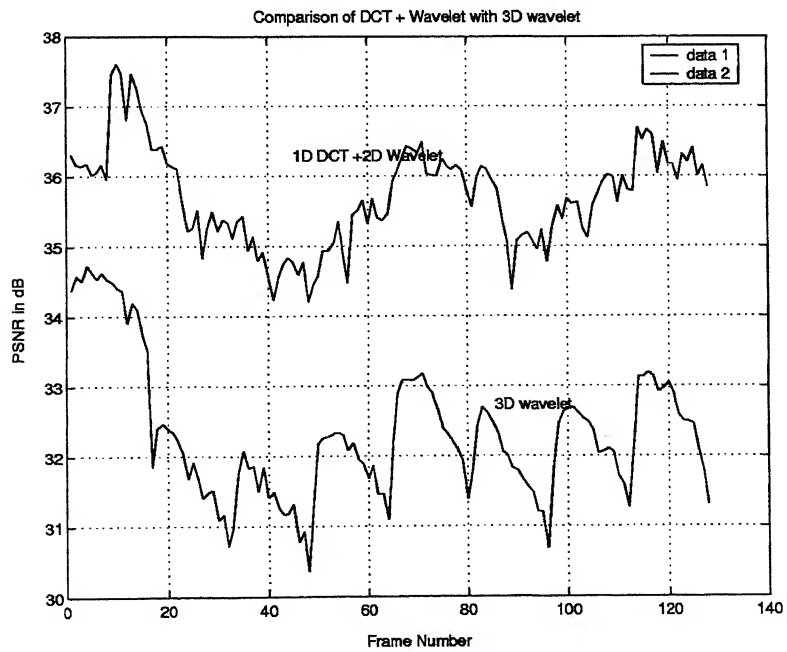


Figure 3.8: Comparison between the Proposed coder and 3D wavelet coder for Hallmonitor sequence at 0.05 bpp

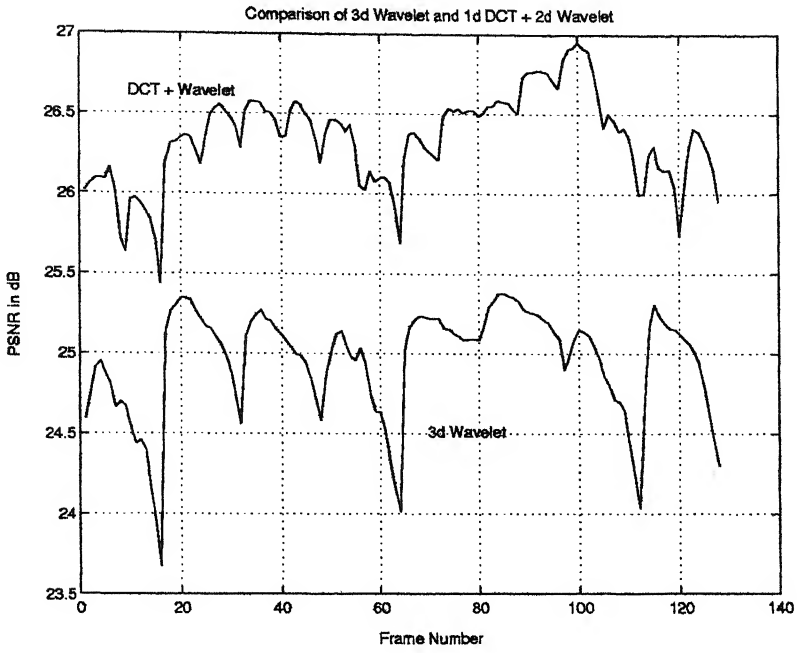


Figure 3.9: Comparison between proposed and 3D wavelet coder for Salesman at 0.01 bpp

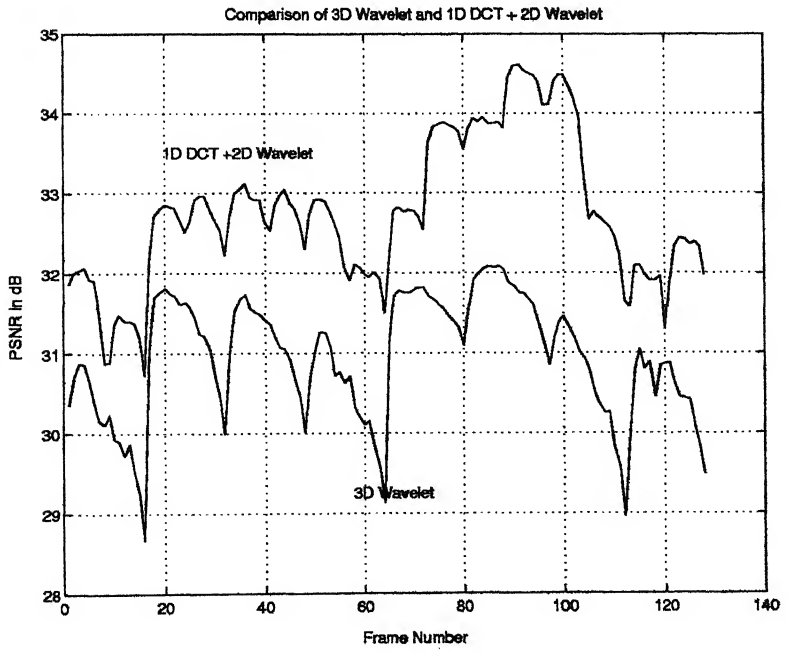


Figure 3.10: Comparison between proposed and 3D wavelet coder for Salesman at 0.05 bpp

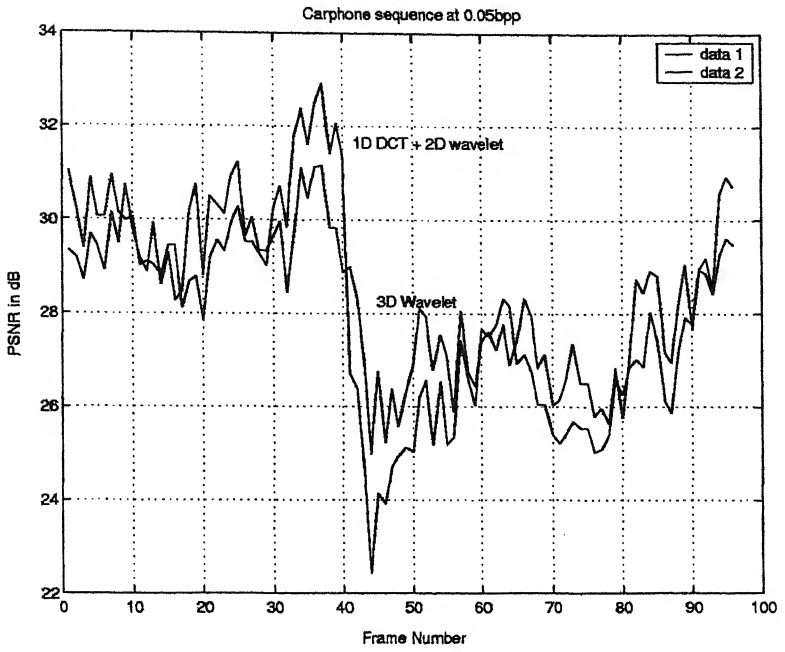


Figure 3.11: Comparison between proposed and 3D wavelet coder for Carman at 0.05 bpp

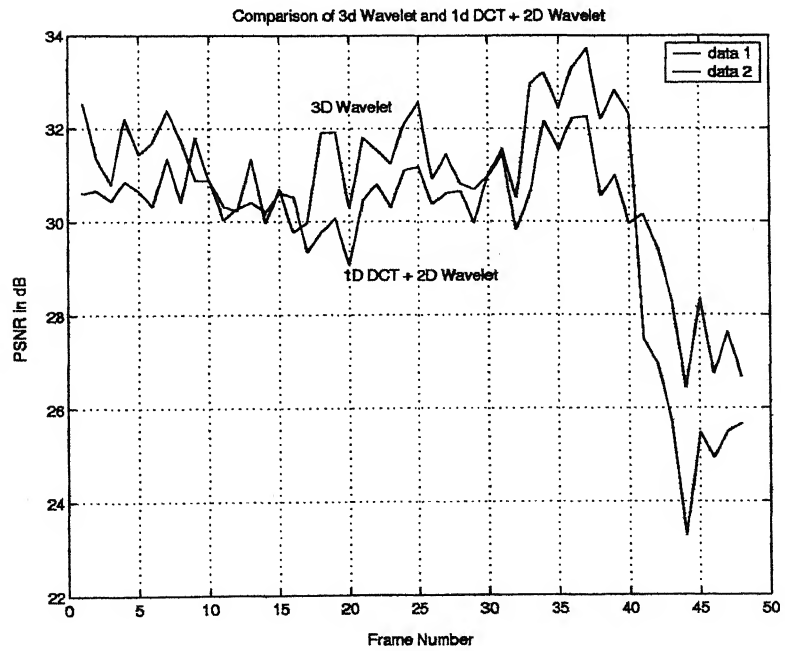
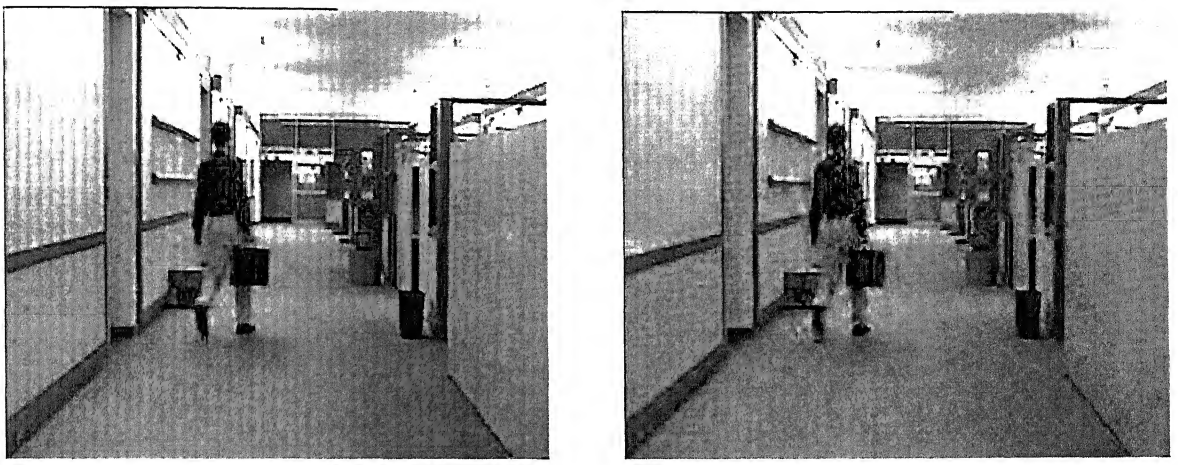
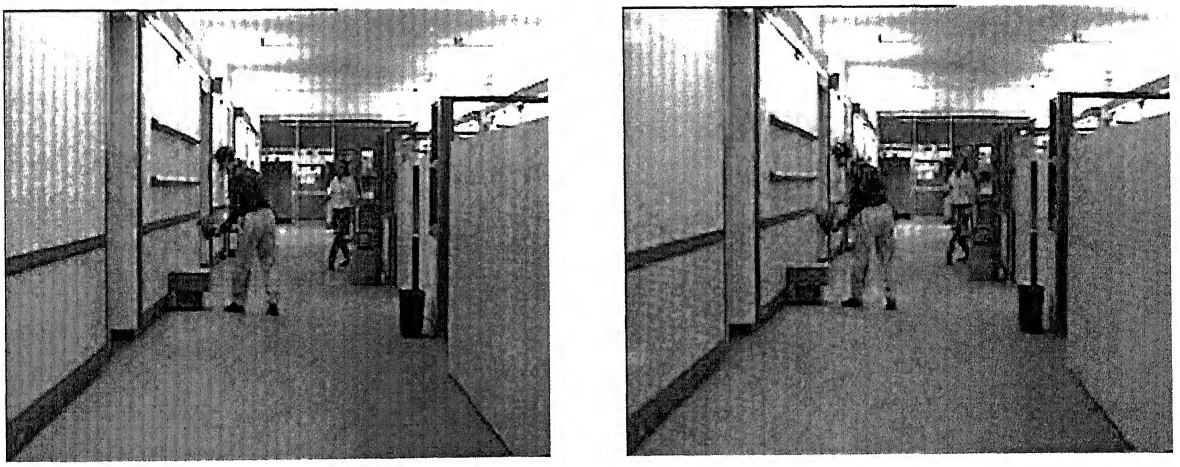


Figure 3.12: Comparison between proposed and 3D wavelet coder for Carman at 0.07 bpp



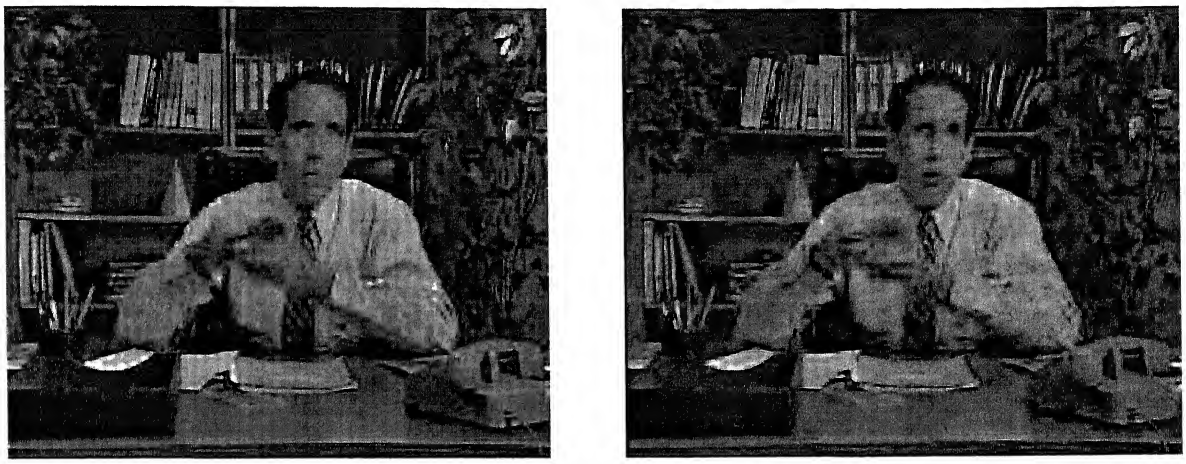
(a) (b)

Figure 3.13: Hall Monitor : 48^{th} Frame a) Proposed Method b) 3D wavelet Coder at 0.05 bpp



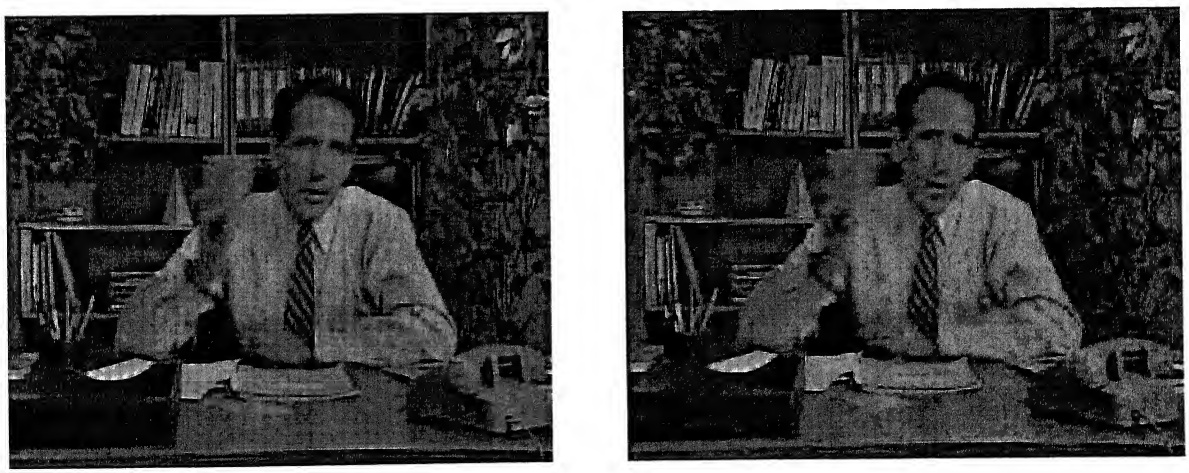
(a) (b)

Figure 3.14: Hall Monitor : 96^{th} Frame a) Proposed Method b) 3D wavelet Coder at 0.05 bpp



(a) (b)

Figure 3.15: Salesman : 16^{th} Frame a) Proposed Method b) 3D wavelet Coder at 0.05 bpp



(a) (b)

Figure 3.16: Salesman : 62^{nd} Frame a) Proposed Method b) 3D wavelet Coder at 0.05 bpp

(a) (b)

Figure 3.17: Carman :14^{th} Frame a) Proposed Method b) 3D wavelet Coder at 0.05 bpp

(a) (b)

Figure 3.18: Carman : 26^{th} Frame a) Proposed Method b) 3D wavelet Coder at 0.05 bpp

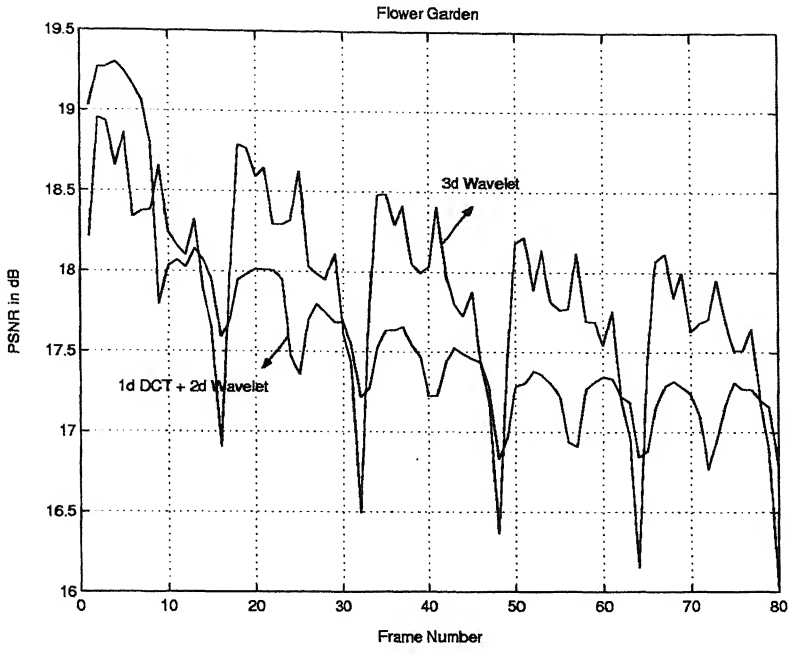


Figure 3.19: Comparison between proposed coder and 3d wavelet of Flowgarden at 0.05 bpp

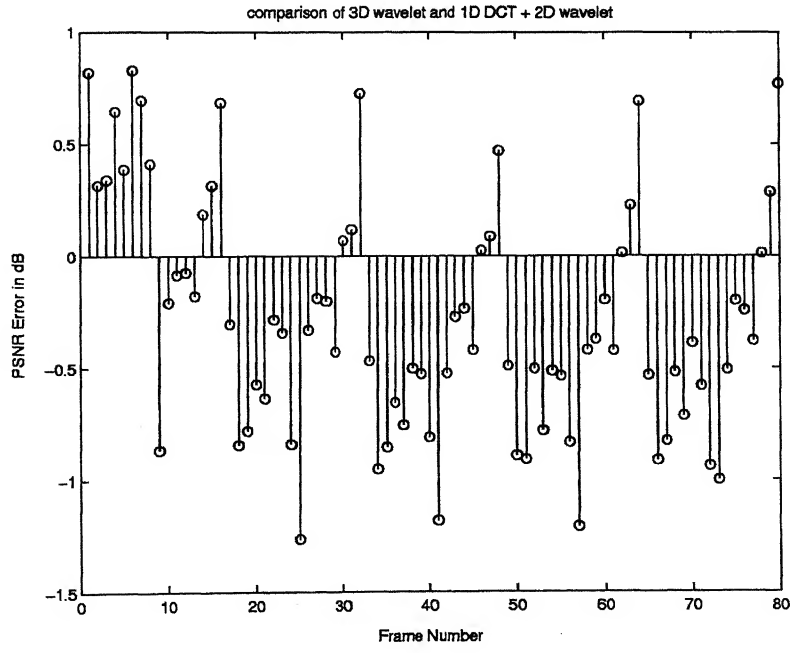


Figure 3.20: PSNR gain/loss of Flowgarden in comparison with 3D wavelet coder frame wise at 0.05 bpp

Chapter 4

Scalable Coding

Now-a-days, video coding is not only confined to compressing spatio-temporal visual information but is also focussed to incorporate additional features like scalabiliy, region of interest coding etc. Moreover, increasing demand of video-delivery over Internet, unreliable wireless-mobile channel has necessitated restructuring of the entire coding strategy so as to produce a bit-stream sufficiently robust against packet loss and burst error type transmission impairments. In this chapter, we shall discuss only the scalability aspect. We will highlight on how scalable coding can be achieved using the proposed codec and leave the issues like region of intrest coding as a part of future investigation.

4.1 Scalability : Why is it needed ?

In the last few decades we have seen the tremendous growth of both Internet and multimedia technologies. Day by day, it is giving birth to new interesting applications and services. Video-on-Demand (VOD), Interactive TV (ITV), Interactive Hypermedia Courseware is just a few among them. Let us consider the video-on-demand application for our discussion. There must be a multimedia video server

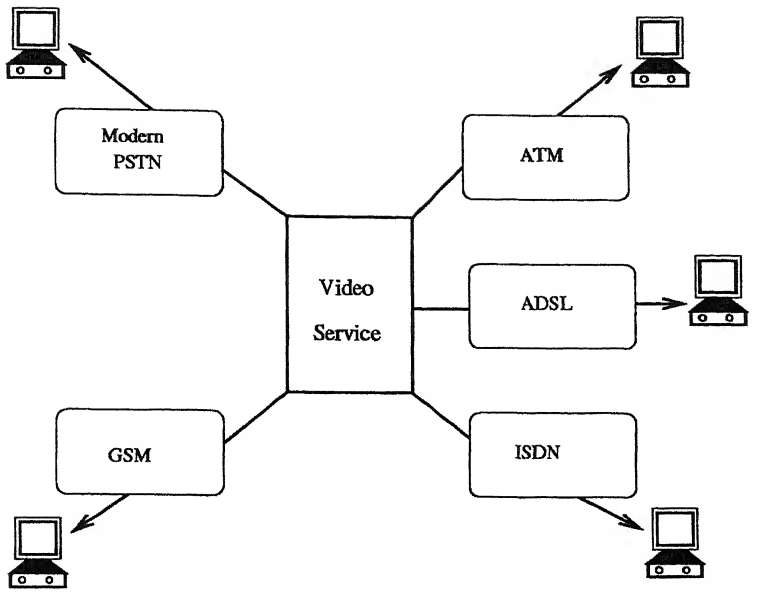


Figure 4.1: A Heterogeneous Communication Network with Video Services.

where all the videos are stored on-line, that is on magnetic disk. Also, there will be thousands of clients/users connected to it through different networks, Internet, demanding for videos. Now, it is quite natural that these users will have wide varieties of computational resources, display and memory capability and screen resolution. Also the available network bandwidth may differ from client to client, time to time. Network reliability (e.g. packet loss characteristics in case of Internet) may fluctuate in random fashion. In such a receiver-driven multicast application in a complicated heterogeneous environment, in order to guarantee the Quality-of-service (QOS) to all the users, the server needs to optimize the video streaming for each of the individual user! However, this task is rather involving and almost impossible to carry out. As an alternative solution, the server can serve one common omnipotent video stream that can simultaneously support all the necessary requirements tailored to individual services.

Scalable video coding, sometimes also called Layered video coding is an approach to materialize the second elegant solution. A conventional method to attain scalability is to split and distribute the video information over a number of layers. The

most fundamental information to reconstruct the video frames is packed into a layer referred to as base layer. The other layers, called enhancement layers, are for additional information which, when combined with the base or the lower enhancement layers , produce more refined information . More the number of layers the decoder receives, the better is the quality it will achieve. To date, standards like H.263, MPEG-2 and MPEG-4 have accepted this method.

A comparatively newer method to support scalability is to produce an embedded bit-stream. An embedded bit-stream can be truncated from the end to match any reduced target bit-rate. Any truncated version of the parent bit-stream can reconstruct back the video. However, the actual quality of the decoded video depends on the point of truncation. The closer is the point of truncation toward the end, the better is the quality. Noteworthy, much of this work was inspired by the design and excellent coding efficiency of the EZW and SPIHT coders for still images. Bits are packed in the order of "significance", a measure that is often tied with the magnitude of image coefficients in the transform domain. Therefore, the image coefficients are ordered first by their magnitude and then packed in a bit-stream in descending order of bit-planes. In EZW and SPIHT the embedding is so fine granular that one can scale the size of a bit-stream in byte accuracy. For this reason, a embedded bit-stream is said to have fine granular scalability (FGS). Embedding a video bit-stream is a simple straightforward extension of embedded image coding if a 3D video transform is used.

Structuring the video bit-stream in this manner helps the individual receiver to select some part of the bit-stream and decode it with its available capabilities to fulfill its own requirement. Also ideally, the different subset bit-streams extracted from the full parent bit-stream provide the maximum possible amount of video information to any user despite its limited computing power, connection bandwidth, and so on. In this way, by indirectly adapting the content made for some high-end machines to

some less powerful machines, the overheads involved in producing different versions of the same information for different clients can be virtually eliminated.

4.2 Scalability : its different forms

There may be many different types of scalability, e.g. temporal resolution (frame rate) scalability, spatial resolution scalability, bit-rate (SNR) scalability, transmission scalability and object scalability to name a few. A system that can handle all of these different scalability aspects would like to be of great importance in near future. However, in our present discussion, we will confine ourselves within theree major kinds of scalability: temporal scalability, spatial scalability and data-rate scalability.

4.2.1 Temporal Resolution Scalability

Temporal resolution (frame-rate) scalability allows the user to adjust the refresh rate of the decoded frames while playing back the video.

In standard hybrid video coding system e.g. in all MPEG family video coding standards and H.263+, temporal scalability is achieved by including B-frames in a compressed video bit-stream. B-frames are bi-directionally predicted from the forward and backward I- of P-frames, and not used as referece pictures for motion compensation in reconstructing other B- or P-frames. For this reason, a B-frame can be dropped without affecting the picture quality of other frames. However, dropping a B-frame will lower the frame rate or temporal quality of video playback, giving the meaning of temporal scalability.

4.2.2 Spatial Resolution Scalability

Spatial resolution scalability is a funcionality to decode frames at different sizes, e.g. from size of thumbnail for browsing to that of HDTV.

Spatial scalability is obtained by creating a multiresolution representation of each frame in a video bit-stream. This multiresolution representation is used to split each frame into set of layers. In this case an increased number of reconstruction layers correspond to higher spatial resolution of the individual frames of the video. It is worth noting that an enhancement layer can also be split into multiple enhancement layers so that each one is built upon the previous ones at a lower resolution.

In MPEG, H.263+, the multiresolution representation is attained in DCT frame-work whereas in case of 3D video coding system, the same is obtained in subband coding/wavelet frame work.

4.2.3 Data-rate Scalability

Data-rate (SNR) scalability helps to attain progressively increasing quality as more and more of the bit-stream is decoded. This particular type of scalability applies to most types of media like video, sound, speech etc. It is generally achieved using layered quantization technique that splits the bit-plane depth into a set of layers.

Finally, it's worth mentioning, a scalable bit-stream does not always have a single type of scalability. In fact, different types of scalability often co-exist in a multidi-mensional structure, so as to provide a wide range of adaptation choices.

4.3 How Spatial and SNR Scalability is achieved with the proposed coder

The proposed video coder naturally gives scalability in rate because of 3D-SPIHT algorithm. However, it is highly desirable to have temporal and/or spatial scalabilities for today's many multimedia applications such as multicast network distributions. We have used Multiresolution encoding to generate layered bit-stream and achieved Spatial and SNR scalabilities. Depending on the bandwidth availability, different combinations of the layers can be transmitted to the decoder to reconstruct the video sequences with different spatial/temporal resolutions. Information bits corresponding to different resolutions are interleaved. Fortunately, the SPIHT algorithm allows us to keep track of the spatial resolutions associated with information bits. Specifically, multiresolution encoding amounts to putting into the first layer (low-resolution) all the bits needed to decode a low-resolution video and the second layer (higher resolution) consists of those bits to be added to the first layer for decoding a higher resolution video sequence and so on.

The layered bit-stream for the proposed coder is generated as follows. After the application of 1D-DCT in the temporal dimension, each of the frequency planes is decomposed using dyadic wavelet decomposition once, resulting in four sub-bands for the first GOF. If it is not the first GOF, the present DC frequency plane is subtracted from the previous DC plane and as in the earlier case one dyadic wavelet decomposition is carried out. From the resultant frame structure, LL_1in all frequency planes correspond to the base layer and the remaining subbands i.e LH_1, HL_1 and HH_1 constitute the single enhancement layer. Then DWT is applied on each of the sub-bands of each frequency plane seperately just as in the case of a wavelet packet, after which 3D-SPIHT is used to transmit each of the base layer and the three layers of the single Enhancement layer.This modified layered bit stream is shown in Figure

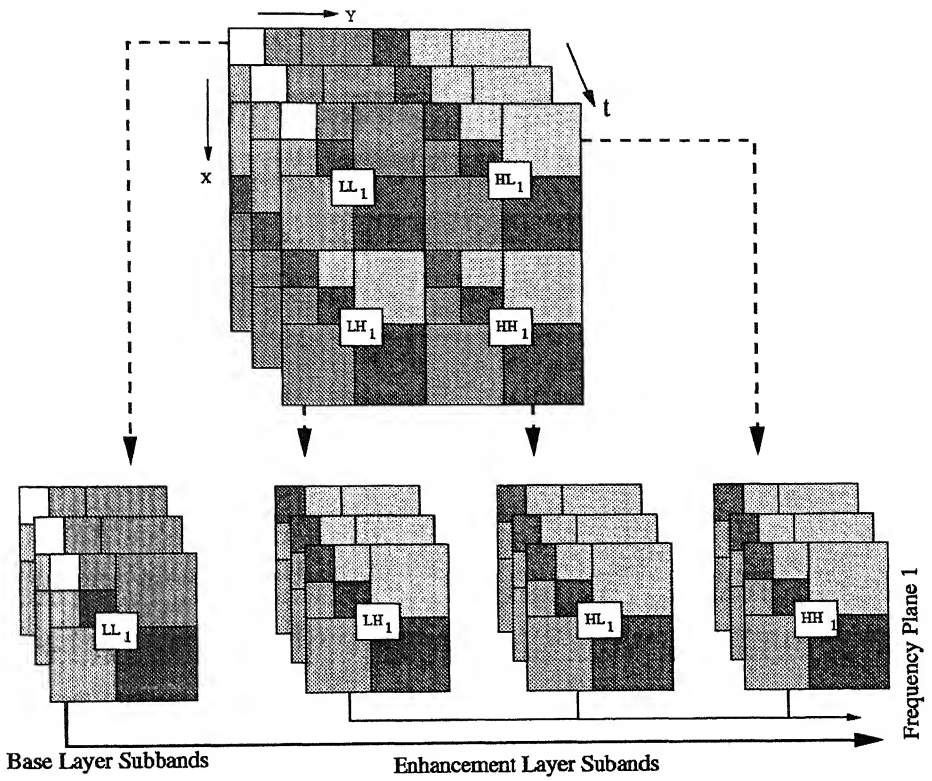


Figure 4.2: Subband Decomposition Required to achieve Spatial and SNR scalabilities.

4.2.

Now, the decoder has two distinct alternatives in reconstructing back the video.

- Firstly, the decoder can choose all the frequency planes, but use only the base layers to construct the video.
- Secondly, the decoder can choose all the frequency planes, take all the enhancement layers in addition to the base layers, but use the truncated bit-stream corresponding to each layer.

In the first case the reconstructed video will be of lower spatial resolution. If the decoder had used the enhancement layer too, it could have had the full spatial resolution.This type of decoding corresponds to*Spatial Scalability.* In the second way of decoding it will give rise to a low quality video which can be used for receivers with limited band-width in the case of a multicast environment. Had the decoder used more number of bits from each bit-stream layer, the decoded quality could have been improved to the corresponding extent. This type of decoding corresponds to*SNR or data-rate Scalability.*

4.4 A Word about Temporal Scalability

While reconstructing, the decoder can choose to use the full bit stream pertaining to a particular frequency plane, and select only the first few of the available frequency planes. In this case, the decoder will reconstruct a lower temporal resolution video. The more is the number of selected frequency planes, the better is the temporal resolution achieved. The idea is quite interesting and coaxing which makes one say that this coder also has the Temporal Scalability. This is so as we consider each freqeuncy plane as a separate temporal subband and attempt to incorporate

temporal scalability in a manner that 3D-SBC has used in the past. This type of decoding actually corresponds to *Temporal Scalability*.

But, we maintain that there is not much to gain with this sort of scalability for the proposed coder because if we select the first four frequency planes out of the available eight, 95% of the bit-budget will be allocated to it by 3D-SPIHT and only 5% to the remaining four AC frequency planes. So, there is not much gain with this sort of scalability in our coder.

4.5 Implementation Details and Results

All the implementation details of the original coder like filter selection, number of frames in a GOF also hold true for the modified bit-stream case. Three levels of dyadic wavelet decomposition are performed frequency plane wise. After the base layer and the enhancement layers are formed, if the coder is given a particular bit-budget, 60% of the bit-budget goes into the base layer and the remaining 40% is equally shared between the three enhancement layers. Figure 4.3 shows the 13^{th} and 54^{th} frames of the decoded Akiyo video sequence coded at 0.05 bpp when only the base layer is received; and Figure 4.4 shows the same frames when both the base and enhancement layers are received. Similarly, Figure 4.5 shows the 155^{th} and 164^{th}frames of the decoded Foreman video sequence coded at 0.05 bpp when only the base layer is received and Figure 4.6 shows the same frames when both the base and enhancement layers are received. These figures show different spatial resolutions of the decoded video sequence. The improved quality of the lower spatial resolution is due to the fact that the SPIHT algorithm extracts first the bits of largest significance in the largest magnitude coefficients which reside in the lower spatial resolutions. At low bit-rates the low resolution will receive most of the bits and be reconstructed fairly accurately, while the higher resolution will receive relatively few bits and be

reconstructed rather coarsely. Figures 4.7 and 4.8 illustrate SNR scalability which is possible because of the embedded coding inherent in SPIHT. SNR scalability is however inherent in the SPIHT algorithm itself.

(a) (b)

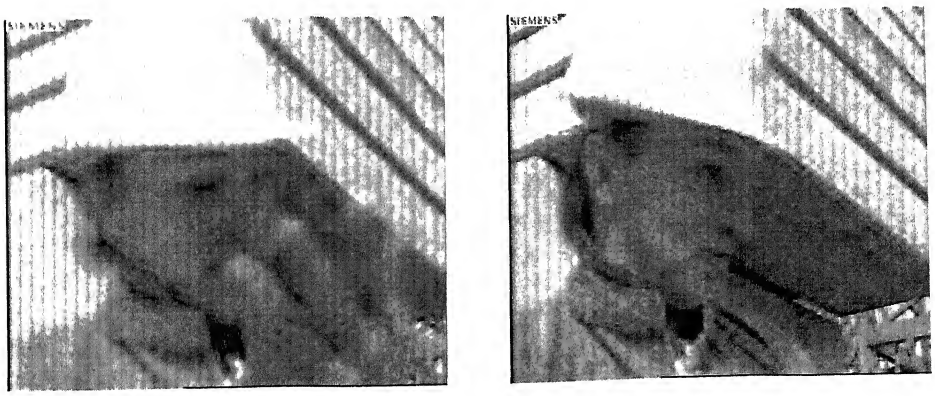
Figure 4.3: Akiyo Sequence 13^{th} and 54^{th}frames : base layer only

(a) (b)

Figure 4.4: Akiyo sequece 13^{th} and 54^{th} Frames : both base and enhancement layers

(a) (b)

Figure 4.5: Foreman Sequence 155^{th} and 164^{th} : base layer only



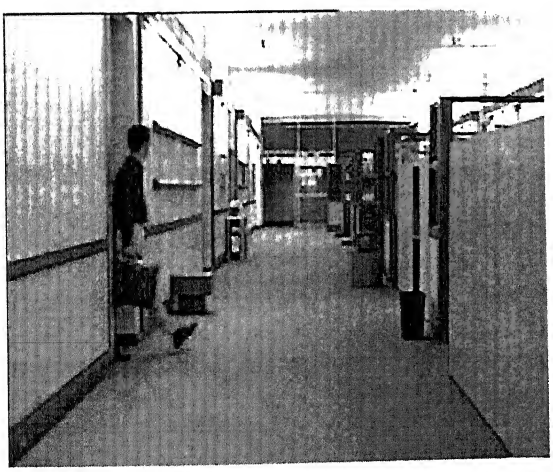
(a) (b)

Figure 4.6: Foreman Sequence 155^{th} and 164^{th} frames : both base and enhancement layers

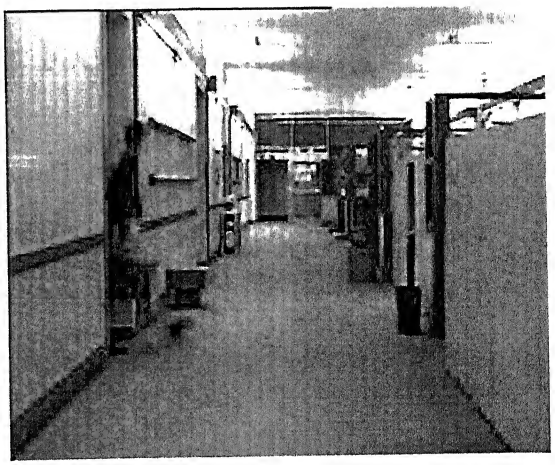
(a) Coded at 0.05 bpp (PSNR = 31.8 db)

(b) Reconstructed at 0.01bpp (PSNR=24 db)

Figure 4.7: Illustration of SNR Scalability : Akiyo, 32^{nd} frame



(a) Coded at 0.05 bpp (PSNR=30.5 db)



(b) Reconstructed at 0.02bpp (PSNR=26.4 db)

Figure 4.8: Illustration of SNR scalability : Hall Monitor, 23^{rd} frame

Chapter 5

Packetization

After coding the video using any method, if it has to be sent over the Internet, the resultant bit-stream must be divided into packets and sent over the network to the destination using different routes. This process of dividing the resultant bit-stream into smaller divisions for error-control and correction is called Packetization.

5.1 Problem Identification & Existing Strategies

Wavelet Zero tree coding techniques developed by Shapiro (EZW) and further refinements by Said and Pearlman (SPIHT) provide very high performance and low complexity compression. However, these algorithms are dependent on the state of the system and highly susceptible to errors. A single bit error could potentially lead to decoder derailment.

For example, in the case of SPIHT algorithm, if an error occurs in the refinement pass, it will affect only that concerned pixel whose refinement is being sent. In contrast, if an error occurs during sorting pass, the resultant bit-stream will be derailed and there are no synchronization points unless synchronization occurs randomly. If the output streams from this algorithm were divided into packets, loss of a single

packet (either due to packet misrouting or due to corrupted bits which might be detected with some error detection scheme) would lead to uncontrolled degradation of the quality. Figures 5.1 and 5.2 show one of the frames of the akiyo video sequence when one and two packets are lost respectively. The point to be observed from these figures is the ungraceful degradation of PSNR with respect to loss of packets. We would prefer an algorithm which gives a graceful degradation with increasing packet losses even at the cost of loosing some compression.

Figure 5.1: Reconstructed frame of akiyo video sequence when one packet is lost

Figure 5.2: Reconstructed frame of akiyo video sequence when two packets are lost

Various strategies exist to address these issues. Retransmission protocols (ARQ) allow the decoder to request the encoder to send a packet again. Forward Error Correction (FEC) techniques and Unequal Error Protection (UEP) allow a certain number of errors to be detcted. Both these strategies come at a cost. Retransmission protocols introduce delay and the retransmitted packets might not arrive soon enough to be useful. Hence, this method cannot be useful for real time transmission. Error correction schemes reduce the compression achieved as extra bits are added. There are a number of existing methods which use this strategy like[22, 23]. Although these algorithms do well to some extent in finding out the errors in the bit-stream they do not guarantee the non occurrence of errors which can potentially derail the coded bit-stream at the decoder.

5.2 Packetization using Packetized Zero tree Wavelet (PZW) Coding

An effective packetization technique is a technique which distributes its bit-stream after encoding in such a way that all the packets contain equal amount of information. It should also be robust to bit errors that occur within the packet or packet losses that take place because of network congestion. Even if bit errors or packet losses occur, the decoder should be able to synchronize back to the encoded bit stream. Furthermore, there should be graceful degradation of video quality as packet losses increase. Keeping all these points in view, we have used PZW coding for packetizing the encoded bit-stream. The details of this algorithm are given in order but the more curious reader may refer to [24, 25].

The encoder begins with the basic SPIHT algorithm. The coefficient tree structure used is the new tree structure discussed in Section 2.4.3. The encoder encodes the video out to a target bit-rate (0.01-0.04 bpp) and stores the bits. Each stored

bit is associated with exactly one of the trees. The SPIHT coder puts out bit streams in which the bits corresponding to different trees are interleaved which yields progressivity. The most significant bit for every coefficient of every tree is made known to the decoder before transmitting any information about the next significant bit. To achieve noise robustness, we sacrifice this progressivity. The output bit-stream is de-interleaved and re-organized into different sub-streams where each sub-stream contains information pertaining to only one tree of coefficients.

A cumbersome but straight forward fixed-length packetization method can now be seen, which serves as an introduction to this method. All the resultant sub-streams are ordered in some fixed order (e.g., a raster scan) known to both encoder and decoder. We assume initially that we are using ATM packets (48-byte payload) and that no sub-stream has more than 48 bytes. The encoder concatenates sub-streams into a packet until no more will fit. If only n trees fit, the encoder pads*out* any space remaining in the packet with null bits, and tree $n+1$ starts the next packet. A substantial amount of overhead would be required in each packet. Firstly, each packet needs to say which tree begins the packet. If the first packet contains trees 1-4, and the second contains 5-11, in the event that the first packet is lost, the decoder would not know that the second packet begins with tree 5. So there should be some overhead bits which indicate which tree starts the packet. Secondly, the decoder must be able to parse out the concatenated sub-streams. At any point in SPIHT algorithm, by interpreting the bits received up to that point, the decoder can determine to which tree the next bit pertains. The decoder simply marches through the decoding threshold levels and the trees (sets) until either a stop code is encountered or a pre-determined target rate is reached. However, when the sub-streams are de-interleaved and concatenated, a separate stop code would be needed to indicate the terminating point of each sub-stream. Alternatively the encoder can inform the decoder how many bits are in each sub-stream and this would equally

enable the decoder to parse them out.

It turns out that the null-padding and the need for stop codes or explicit bit counts for the trees can all be avoided. By re-interleaving the set of sub-streams in any one packet, the decoder can decode each tree in the packet without stop codes or additional information regarding each tree size. A small piece of additional overhead (say, 4 bits) is required in order to tell the decoder how many trees are interleaved in the current packet. The decoder would be unable to correctly cycle through the round-robin of interleaved trees in the packet if it did not know how many trees there are.

Each packet gets filled exactly. To fill the next packet, the encoder examines the upcoming ordered sub-streams. Suppose the next n trees would underfill the packet, the encoder can grow them out by encoding them at a rate higher than the initial target rate. As necessary, more sorting and refinement passes are conducted for those trees alone, and the results interleaved, until the packet is exactly filled. Alternatively, the $n+1$ trees overfilling the packet can be pruned back until the packet is filled exactly. The encoder currently chooses between growing a smaller set of trees and pruning a larger set by taking whichever is closer to 48 bytes, but the decision could be based on a distortion-rate trade-off, or by using lookahead to see how well future groups of trees will fit into future packets.

Growing and pruning lead to spatially varying video quality. Coefficients are not all coded down to the same bit plane (threshold). Each packet has its own termination threshold which is not told to the decoder. The decoder reads a packet by first reading the packet header (which tells how many trees are in the packet, and where the first one is located spatially) and then repeatedly cutting the threshold in half and marching through the bit planes until it reaches the end of the packet. Because trees are grown or pruned to fit within the fixed length segment, no trees span across packet boundaries. Together with the small packet headers, this makes each packet

independently decodable and provides robustness against packet loss.

In transmission over packet switched networks, packets which are not received within a specified time because of network traffic or misrouting are considered "lost". Those packets will not be available to the decoder. Also, a packet may be discarded by the decoder (considered lost) if errors are found in the packet payload after arrival. By dedicating 16 bits of the payload to a CRC, the decoder could have a high probability of error detection over the payload. However we do not use any CRC in the present implementation. Because of the packet independence, in the event of losses, the decoder fills in as many positions as it can in the wavelet coefficient array using all successfully received packets, missing wavelet coefficients are then replaced by zeros. If errors were not detected, the wavelet coefficients corresponding to trees in the erroneous packets would not be replaced by zeros, but would be placed with wrong values in the array; nonetheless the undetected errors cannot propagate beyond the packet boundary and trees of coefficients in other packets would be unaffected.

5.3 Implementation Details & Results

5.3.1 Implementation details

The performance of the same basic algorithm can be quite different according to the actual implementation. Thus, it is necessary to sepcify the main features of the implementation in detail.

- A CIF (352X288) resolution video at a frame rate of 30 fps. The number of frames in a GOF were taken to be 8. We have used 4 levels of decomposition for spatial dyadic wavelet decomposition for each frequency plane which results in 396 (18x22) head coefficients and thereby 396 sub-streams after SPIHT

coding. We have tested this packetization method for various low bit-rate video sequences like "Akiyo", " Hall Monitor", " Coastguard" and " Salesman".

- The filters used for Spatial wavelet decomposition are " Daubechies 2" filters commonly called 'db2' in Matlab. These filters are used although they give less compression compared to " Daubechies 9/7" filters because of their simplicity.

- We have used standard ATM packets which contain 53 bytes out of which 48 bytes are given for payload and other 5 bytes as network routing header. So, the header information regarding the coded video will be within these 48 bytes. The packet header consists of 9 bits representing spatial location of each tree and another 5 bits for the number of trees present in each packet.

- In case if n trees underfiil a packet, decision on whether to include the next tree or use null padding is made as follows. If the 80% of the$n+1$ th tree comes within the packet bit-budget then the packet will also add the next tree. In case if this does not happen the packet will be null-padded.

- For the sake of stop code which is used to seperate between each tree, we have used 7 bits indicating the length of the tree. If the length of the tree is more than 127 an escape word 1111111 is used followed by the code word of 9 bits which indicate the length of the tree.

- The effects of a lost packet can be mitigated by interpolation. We used a simple averaging. The decoder interpolates missing coefficients in all the bands by averaging together as many neighbors as available.

- Two kinds of errors have been investigated : Random packet errors and Burst packet errors. In the case of Random packet errors we randomly select the packets that are lost and calculate the resultant PSNR without error concealmet and with error concealment (i.e after averaging has been done) at different

percentages of packets lost (2%, 5%, 10%, 15%). Burst packet errors occur in case of networks which have a lot of congestion. We actually select a particular packet and then say few consecutive packets after it are lost depending on what percentage of Burst packet errors we are investigating. These errors are also investigated both before and after Error Concealment. PSNR is calculated using the same equation given earlier for the case of original coder (Eq.3.1).

5.3.2 Results

Figures 5.3, 5.4, 5.5 and 5.6 show one of the frames of the Salesman video sequence at 0.04 bpp when it is subjected to different percentages of random packet losses before and after Error-concealment respectively. Figures 5.7, 5.8, 5.9 and 5.10 show one of the frames of the Akiyo video sequence at 0.04 bpp when it is subjected to different percentages of Burst packet losses before and after Error-concealment. Tables 5.1 and 5.2 show the mean PSNR of "Salesman", "Container" and "Akiyo" with different percentages of Random packet errors before and after error concealment. Tables 5.3 and 5.4 show the same information but for the case of Burst packet errors. The results show a graceful degradation of PSNR with increasing packet losses.

Video	BPP	0%	2%	5%	10%	15%
Akiyo	0.04	27.86	23.03	19.47	17.05	14.65
Salesman	0.04	28.14	24.88	20.69	18.36	17.12
Container	0.04	26.26	21.11	15.37	13.30	12.14

Table 5.1: Mean PSNR at different percentages of random packet losses without Error Concealment

Video	BPP	2%	5%	10%	15%
Akiyo	0.04	27.16	26.05	21.58	19.61
Salesman	0.04	27.60	26.66	24.67	23.09
Container	0.04	25.80	23.05	22.37	21.21

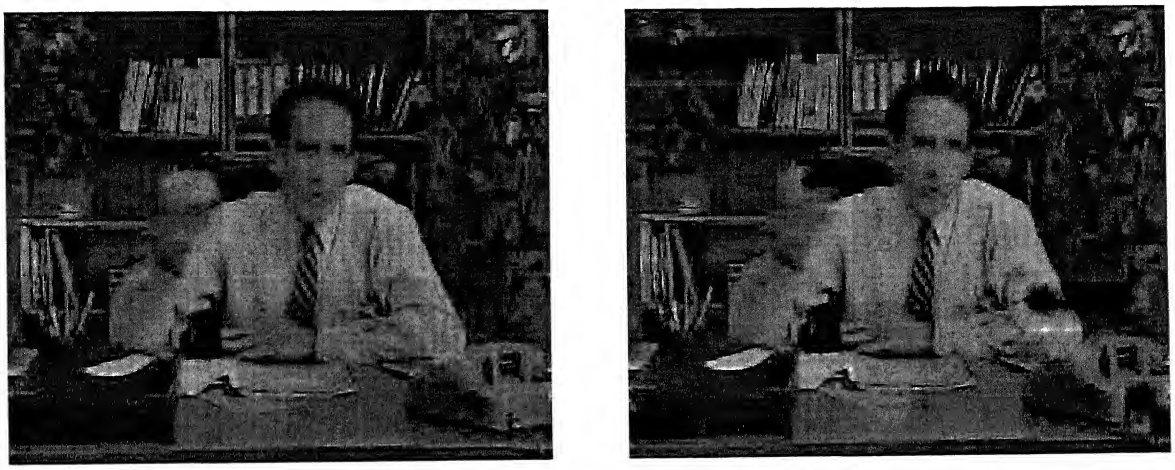
Table 5.2: Mean PSNR at different percentages of random packet losses after Error Concealment

Video	BPP	0%	2%	5%	10%	15%
Akiyo	0.05	32.50	26.90	19.11	17.58	15.65
Salesman	0.04	28.14	26.82	23.78	22.30	19.89
Container	0.04	26.26	24.73	14.24	12.69	12.45

Table 5.3: Mean PSNR at different percentages of Burst packet losses without Error Concealment

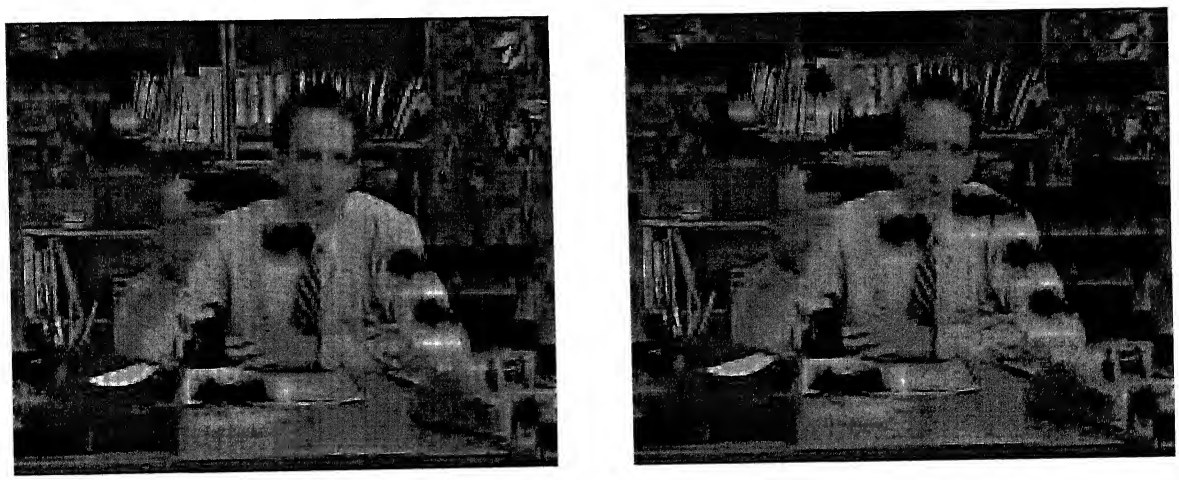
Video	BPP	2%	5%	10%	15%
Akiyo	0.05	31.25	28.13	25.82	22.36
Salesman	0.04	27.83	26.85	26.44	25.17
Container	0.04	25.83	25.27	24.09	22.02

Table 5.4: Mean PSNR at different percentages of Burst packet losses after Error Concealment



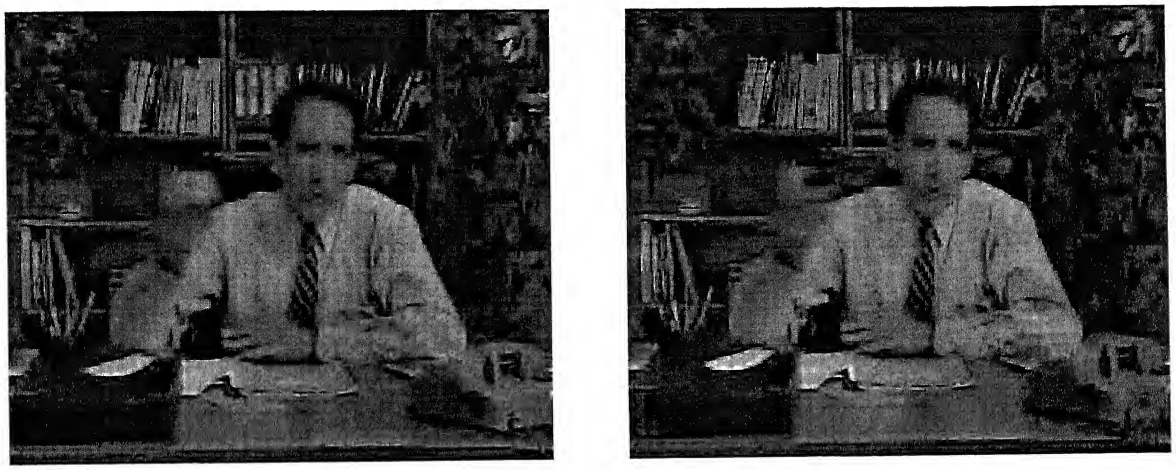
(a) (b)

Figure 5.3: Reconstructed frame of "Salesman" after 0% and 5% lost packets (random) without Error-concealment



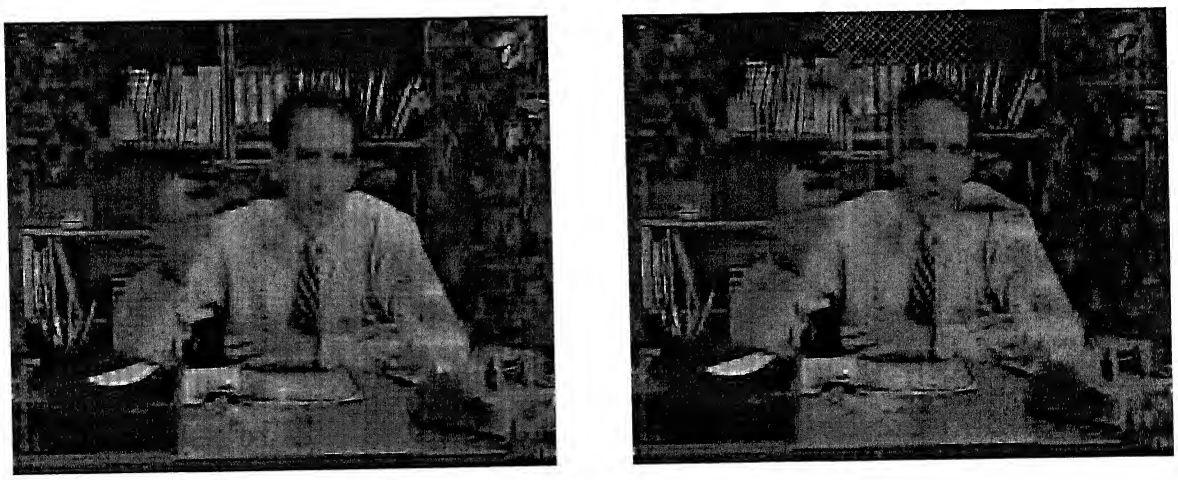
(a) (b)

Figure 5.4: Reconstructed frame of "Salesman" after 10% and 15% lost packets (random) without Error-concealment



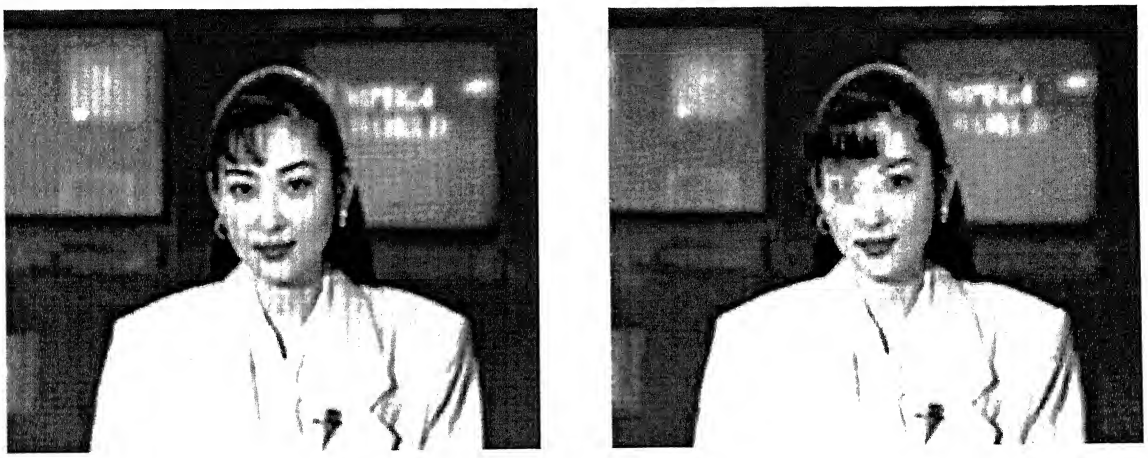
(a) (b)

Figure 5.5: Reconstructed frame of "Salesman" with 0% and 5% lost packets (random) after Error-concealment



(a) (b)

Figure 5.6: Reconstructed frame of "Salesman" with 10% and 15% lost packets (random) after Error-concealment

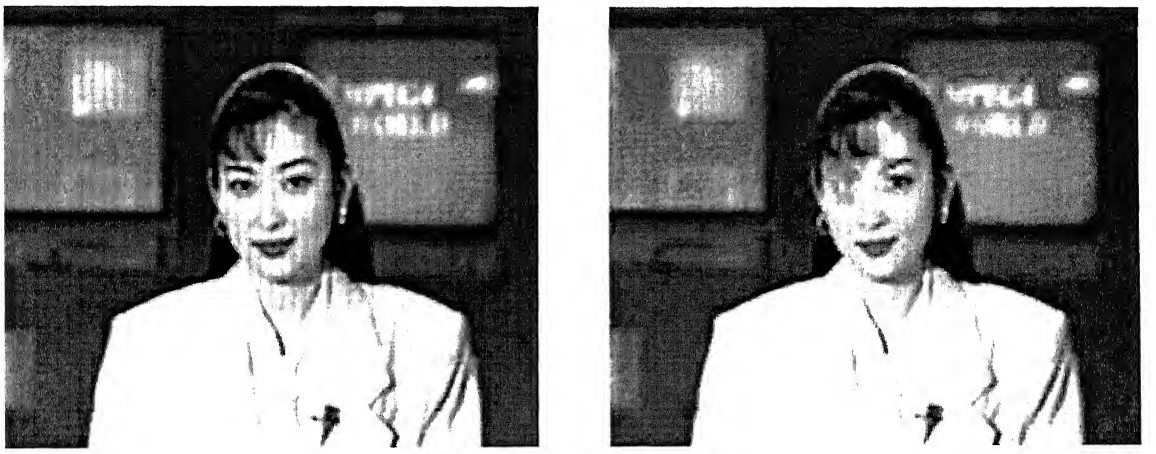


(a) (b)

Figure 5.7: Reconstructed frame of "Akiyo" after 0% and 2% lost packets (burst) without Error-concealment

(a) (b)

Figure 5.8: Reconstructed frame of "Akiyo" after 5% and 10% lost packets (burst) without Error-concealment



(a) (b)

Figure 5.9: Reconstructed frame of "Akiyo" with 0% and 2% lost packets (burst) after Error-concealment

(a) (b)

Figure 5.10: Reconstructed frame of "Akiyo" with 5% and 10% lost packets (burst) after Error-concealment

Chapter 6

Conclusion and Further Scope

In this thesis we proposed a 3D video coder which used Mixed transforms and 3D-SPIHT with a new tree structure in terms of parent-offspring relationships. We have also added the advanced features of Scalability and Effective packetisation which satisfied our objective of making the coder a "complete" one. Spatial and SNR Scalabilities were achieved modifying the original bit-stream to generate a layered bit stream. The packetisation was done using PZW coding algorithm. It showed robustness against both Random packet losses and Burst packet losses and resulted in a graceful degradation of PSNR with increasing packet losses. This was not quiet possible if we had packetised the original bit-stream as it is. Performance comparisons of this coder were made with a coder using 3D-wavelet and 3D-SPIHT and satisfactory results were obtained. We do not have an overall say about the performance of proposed coder in comparison with video coders using Motion compensation. We strongly feel that this coder should not be bargained for coders using motion compensation which provides marginal gains in PSNR at the cost of high computational complexity. The attractive features of the proposed coder include Simplicity, Scalability, Effective packetisation robust to packet losses and no Motion compensation. The coder has the potential to be used for Video conferencing and

Video monitoring applications.

One disadvantage that the proposed coder has is that it does not work well for Video sequences in which the background changes rapidly. Some sort of background correction for the existing coder can be taken up as a part of future work. Throughout the work not much attention was paid to optimize the proposed algorithms. The primary goal was to test whether the ideas proposed indeed worked. Rigorous optimizations of different parts of these algorithms can also be taken up as a part of future work.

Bibliography

[1] ITU SG 16 Q.15, " H.26L Test Model Long Term Number 3 (TML-d),"Doc. Q15J08, Osaka, Japan May 2000.

[2] Yui-Lam Chan and Wan-Chi Siu, "Variable Temporal Length 3D Discrete Cosine Transform Coding," IEEE Tranasactions on Image Processing, Vol. 6, No.5, pp. 758-763, May 1997.

[3] G. Karlsson and M.Vetterli, "Three dimesional subband coding of video," *Proc.Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*,pp. 1100-1103, April 1988.

[4] C.I.Podilchuk, N.S. Jayant and N.Fravardin, "Three dimensional subband coding of video," *IEEE Transactions on Image Processing*,vol.4, No.2, pp. 125-139, February 1995.

[5] T.Kronander, "New results on three dimensional motion compensated subband coding," *Proc. Picture Coding Symposium(PCS90)*,March 1990.

[6] J.R. Ohm, "Three dimensional subband coding with motion compensation," *IEEE Transactions on Image Processing*,vol.3, No.5, pp. 559-571, September 1994.

[7] S. J. Choi and J. W. Woods, "Motion COmpensated 3D Subband coding of video," *IEEE Transactions on Image Processing,*vol.8, No.2, pp. 155-167, February 1999.

[8] V. M. Bove and A. B. Lippman, "Scalable open architecture television," SMPTE *(Society of Motion Picture and Television Engineers) Journal,* pp. 2-5, January 1992.

[9] D.Taubman and A. Zakhor, "Multirate 3D subband coding of video," *IEEE Transactions on Image Processing,* vol.3, No.5, pp. 572-*588*, September 1994.

[10] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing,*vol.41, pp. 3445-3462, December 1993.

[11] A. Said and W.A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Image Processing,* vol. 5, pp. 1303-1310, September 1996.

[12] Y. Chen and W. A. Pearlman, "Three dimensional subband coding of video using zero tree method," *proc. SPIE 2727 Visual Communications and Image Processing 96,* pp. 1302-1309, March 1996.

[13] J. Y. Tham, S. Ranganath and A. A. Kassim, "Highly scalable wavelet based video codec for very low bit rate environment" *IEEE J. Select. Areas Commn.,*vol. 16, pp. 12-27, January 1998.

[14] B. J. Kim, Z. Xiong and W. A. Pearlman, " Low bit rate scalable video coding with 3D set partitioning in hierarchical trees (3D-SPIHT)," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 10, No.8, pp.1365-1374, December 2000.

[15] Mika Helsingius, Pauli Kuosmanen and Jaakko Astola, "Image Compression using Multiple transforms," *Image Communication,* August 1997.

[16] Sungadae Cho and William A. Pearlman, "A Full- Featured, Error resilient, Scalable Wavelet video codec based on SPIHT coding algorithm," *IEEE Transactions on circuits and systems for Video Technology,* vol. 12, No. 3, pp. 157-171, March 2002.

[17] Wasfy B. Mikhael and Albert P. Berg, "Image Representation using Non orthogonal Basis Images with Adaptive Weight Optimization," *IEEE Signal Processing Letters,*Vol. 3, No. 6, June 1996.

[18] Anil K. Jain, "Fundamentals of Digital Image Processing," New Delhi, 2001, Prentice Hall of India.

[19] B. B. Chai, J. Vass and X. Zhuang, "Significance linked connected component analysis for wavelet image coding," *IEEE Transactions on Image Processing,*Vol. 8, No. 6, pp. 774-784, June 1999.

[20] D. Taubman, "High Performance Scalable image compression with EBCOT," *IEEE Transactions on Image Processing,*Vol. 9, pp. 1158-1170, July 2000.

[21] Kausik Maiti, "3D Video Coding : A Novel Approach," M. Tech thesis, Indian Institute of Technology, Kanpur, April 2003.

[22] Yao Wang and Qin Fan Zhu, "Error Control and Concealmet for Video Communication : A Review," *Proceedings of the IEEE,* vol. 86, No. 5, pp. 974-997, May 1998.

[23] Joohee Kim, Russell Mersereau and Yucel Altanbasak, "Error Resilient Image and Video transmission over the Internet using Unequal Error Protection," *IEEE Transactions on Image Processing,* Vol. 12, No. 2, February 2003.

[24] Jon K. Rogers and Pamela C. Cosman, "Robust Wavelet Zero Tree Image compression with fixed-length Packetization", Data Compression Conference 1998.

[25] Jon K. Rogers and Pamela C. Cosman, "Wavelet Zerotree Image Compression with Packetization." *IEEE Signal Processing Letters,*vol. 5, No. 5, May 1998.